ANALYTICAL DETERMINATION OF A HELICOPTER HEIGHT VELOCITY DIAGRAM

THESIS

Michael Justin Harris, Major, USMC

AFIT-ENY-MS-18-M-260

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENY-MS-18-M-260

ANALYTICAL DETERMINATION OF A HELICOPTER HEIGHT VELOCITY DIAGRAM

THESIS

Presented to the Faculty

Department of Aeronautical Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Aeronautical Engineering

Michael Justin Harris, B.S.

Major, USMC

22 March 2018

AFIT-ENY-MS-18-M-260

ANALYTICAL DETERMINATION OF A HELICOPTER HEIGHT VELOCITY DIAGRAM

Michael Justin Harris, B.S.
Major, USMC

Committee Membership:

Dr. Donald Kunz
Chair

Capt Joshuah Hess, PhD
Member

Dr. Richard Cobb
Member

Abstract

A helicopter height velocity (HV) diagram was analytically constructed using optimal control techniques. A three degree of freedom, point-mass, dynamic model was developed and validated with flight test data. An induced velocity calculation was incorporated which addressed the affects of vortex ring state. The problem was posed as an open final time, constrained initial state, constrained final state problem, with the objective function as a weighted sum of the initial altitude and quadratic controls. This formulation was solved using direct pseudo-spectral collocation and adaptive mesh refinement as implemented by the GPOPS-II® software suite. Proper adjustment of path constraints was crucial in achieving solutions which were comparable with flight test data. Results compare favorably with flight test data and previous analytical HV diagrams.

Acknowledgments

A thousand thanks to Dr. Kunz for his instruction, patience with me, and passion for rotorcraft engineering. His weekly guidance and insight into all aspects of this project were invaluable. For every problem which I brought him he consistently returned a wise solution approach.

This research would have never progressed beyond a helicopter model without the instruction and mentoring of Capt Hess. His expert grasp of optimal control and pseudo-spectral collocation was instrumental to my own understanding and application.

Additionally, I am very grateful to Dr. Cobb for outstanding instruction in the discipline of controls engineering and for his attention to detail and experienced advice as a committee member.

There is one woman who entirely enabled this endeavor. My lovely, devoted wife accomplished all the hard work of running a household and raising our children while her absent husband sat in a quiet and peaceful place, researching or coding. Thank you babe for you faithful love and constant support. I love you!

Lastly, Soli Deo Gloria.

Michael Justin Harris

Table of Contents

Page

Page

Page

Page

List of Figures

List of Tables

List of Acronyms

FAA Federal Aviation Administration

HV Height-Velocity

LGR Legendre Gauss Radau

NAVAIR Naval Air Systems Command

NLP nonlinear programming

OEI one engine inoperable

RTOC Real Time Optimal Control

SGRA Sequential Gradient Restoration Algorithm

SNOPT Sparse Nonlinear Optimizer

SQP Sequential Quadratic Programming

VRS vortex ring state

List of Symbols

α	Rotor Tip Path Plane Angle with Horizon
$\bar{c}_d$	Average Blade Drag Coefficient
$\bar{c}_d$	Average Blade Drag Coefficient
η_{CBOX}	Combining Gearbox Efficiency
η_{MR}	Main Rotor Gearbox Efficiency
η_{TR}	Tail Rotor Gearbox Efficiency
λ	Inflow Ratio
λ_{iJ}	Rotor Inflow, Non-Dimensionalized with v_h
λ_i	Rotor Inflow, Non-Dimensionalized with ΩR
μ	Advance Ratio
μ_z	Non-Dimensional Rotor Velocity, Parallel to Tip Path Plane
μ_z	Non-Dimensional Rotor Velocity, Perpendicular to Tip Path Plane
Ω	Rotor Angular Velocity
Ω_0	Nominal Rotor Angular Velocity
Φ	Terminal Cost
Ψ	Terminal Constraints
ρ	Air Density
σ	Solidity Ratio
τ	Non-Dimensional Time
τ_{fguess}	Guess for final non-dimensional time
θ	Flight Path Angle
A	Rotor Area
A_{cuff}	Rotor Area of the Blade Cuffs
C_P	Power Coefficient
C_Q	Torque Coefficient
C_T	Thrust Coefficient
C_T	Thrust Coefficient
$C_{P_{MR}}$	Main Rotor Power Coefficient
C_{Pc}	Climb Power Coefficient
C_{Pi}	Induced Power Coefficient
C_{Po}	Profile Power Coefficient
C_{Pp}	Parasite Power Coefficient
$C_{T_{TR}}$	Tail Rotor Thrust Coefficient
C_{TX}	Horizontal Component of Thrust Coefficient
C_{Tx}	Horizontal Component of Thrust Coefficient
C_{Tz}	Vertical Component of Thrust Coefficient
C_{Tz}	Vertical Component of Thrust Coefficient
D	Drag
f	Helicopter Dynamics, Path Constraints
f_v	Vertical Drag From Downwash Factor
f_w	Washout Factor for Vertical Drag from Downwash
f_{eR}	Equivalent Flat Plate Drag Area Under the Rotor
f_{ex}	Horizontal Flat Plate Drag
f_{ez}	Vertical Flat Plate Drag
G	Engine Digital Electronic Control Unit Gain
g	Gravitational Acceleration
I_R	Rotor Polar Moment of Inertia

J	Performance Measure
k_G	Ground Effect Factor
k_i	Induced Velocity Correction Factor
L	Running Cost
l_{TR}	Distance From Main Rotor Mast to Tail Rotor Hub
m	Aircraft Mass
P	Engine Power
P_R	Power Required
P_{2AG}	Engine 2 Governed Power Available
P_{2G}	Non-Dimensional Engine 2 Governed Power Available
P_{Acc}	Accessory Power
P_{OEI}	Power Limit for One Engine Inoperable Flight Condition
P_{R1}	Non-Dimensional Helicopter Power Required
$P_{R_{TR}}$	Tail Rotor Power Required
Q_{MR}	Main Rotor Torque
Q_{Req}	Torque Required
R	Main Rotor Radius
t	Time
T_{IGE}	Thrust In Ground Effect
T_{MR}	Main Rotor Thrust
T_{OGE}	Thrust Out of Ground Effect
t_{ref}	Reference Time, $\frac{100}{\Omega_0}$
T_{TR}	Tail Rotor Thrust
tau_1	Engine 1 Time Constant
tau_2	Engine 2 Time Constant
u_j	Controls
v_h	Rotor Inflow in a Hover
W	Aircraft Weight
x_i	States
x_{1maxg}	Guess for max value of x_1 state
x_{9f}	Terminal constraint for x_9
x_{9lop2}	Phase 2 lower bound for x_9
z	Rotor Height Above Ground Level
h	Altitude Above Ground Level
T	Thrust
u	Horizontal Velocity
V	Aircraft Velocity
w	Vertical Velocity, (Positive Down)
x	Horizontal Position

ANALYTICAL DETERMINATION OF A HELICOPTER HEIGHT VELOCITY DIAGRAM

I. Introduction

1.1 Research Overview

If a helicopter loses all engine power, a maneuver called an autorotation is used to descend and safely land the aircraft. Many helicopters utilize two engines to mitigate the chance of complete power loss, but at certain conditions, single-engine flight may also result in an involuntary descent. The combinations of altitude (above the ground) and airspeed which provide insufficient energy to land the aircraft are depicted as a Height-Velocity (HV) diagram. The analytical determination of the low speed HV diagram using optimal control theory is the subject of this research. The present chapter outlines background information on the HV diagram, a basic description of the case study helicopter, research objectives and a thesis overview.

1.2 Single-Engine Flight

Most military helicopters incorporate two engines to reduce the chance of a complete loss of power. Complete power loss is relatively uncommon, but can occur due to combat damage, fuel starvation, fuel contamination, fire, or in some designs, driveshaft failure. More probable is the failure of a single engine, where the power available to drive the main rotor is approximately halved. Common causes of engine failure include fuel contamination, fuel starvation, material failure of a turbine component, compressor stall or foreign object ingestion. Ideally, in the case of single-engine fail-

ure, the remaining engine provides sufficient power for all flight regimes. However, for many helicopters, single-engine failure results in sufficient power at certain airspeeds but a power deficit at other airspeeds.

A helicopter has a power bucket like the drag bucket of a fixed wing aircraft: at high airspeeds, the increased drag on the airframe and rotor demands higher power, and at low airspeeds, the re-ingestion of rotor tip vortices in the flow field also increases the power required by the main rotor. Hence, after failure of a single engine, the aircraft is flown at the most efficient airspeed. For landing however, the helicopter must decelerate to an airspeed where power required exceeds power available. This power deficit is overcome by initiating a rapid descent and using the vertical component of the helicopter velocity as an energy source to drive the main rotor, keeping it turning at a useful angular velocity.

Therefore, the entry flight conditions for a single-engine failure may greatly influence the outcome of a single-engine landing. For a particular gross weight and air density, these entry flight conditions are functions of height above the ground and airspeed, combinations of potential and kinetic energy which can be used to effect a single-engine landing. Certain combinations of gross weight, air density, airspeed and altitude provide insufficient energy for landing the aircraft within structural limitations.

1.3 The Single-Engine HV Diagram

It is common practice to publish a graphical depiction of unsafe flight conditions as an HV diagram. Combinations of unsafe altitude and velocity are charted for a particular air density and aircraft gross weight. An example HV diagram is shown in Figure 1.1. A complete HV diagram depicts both a low speed and a high speed unsafe region, but this research was only concerned with the low speed portion of the

diagram. Henceforth, the term HV diagram refers to the low speed portion only.

The combinations of altitude and airspeed which lack sufficient energy for a partial power landing are shaded and labeled as an unsafe or avoid region. For a twin-engine helicopter, an HV diagram is needed for loss of a single engine and loss of both engines, since the same concept of insufficient energy for landing also applies to an autorotation following complete power loss. The first purpose of the HV diagram is to inform the pilot. The HV diagram does not preclude operations in the unsafe region, but advertises the severe consequences of a single or dual engine failure in certain flight conditions. The second purpose of the HV diagram is in the design of terminal flight profiles. Takeoff and landing profiles are generally standardized to remain clear of the unsafe flight conditions during normal operations, so the HV diagram has a significant impact on take-off and landing profiles for airports, oil rigs, ships and rooftop landing pads. Determination of the HV curves is commonly completed analytically and the analytical solution is validated by flight test. Obviously, flight test in which simulated engine failures determine the boundary between safe and unsafe flight regimes involves inherent risk. Given this risk, and since flight test is expensive by nature, the analytical solution is often validated at a few points and then extrapolated to many different conditions. Hence, greater fidelity is continually sought for analytical construction of the HV diagram.

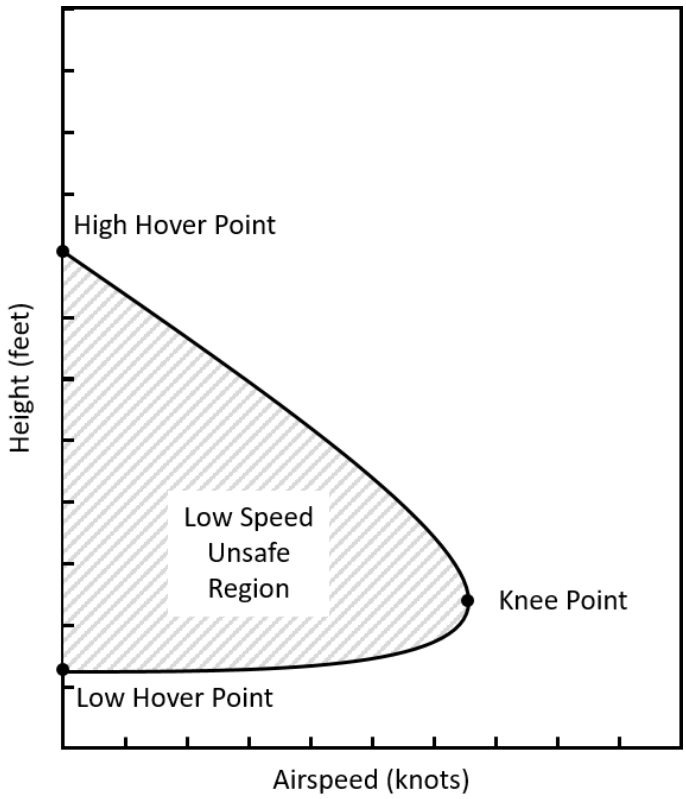


Figure 1.1. Representative Height Velocity Diagram

1.4 Research Catalyst

Many helicopter manufacturing companies have developed analysis programs to analyze performance, stability and control aspects of rotorcraft flight. Among other calculations, these programs are used to conduct analytical evaluations of the HV region. The Naval Air Systems Command (NAVAIR) contains program offices which manage all aspects of Navy and Marine Corps aircraft including acquisition, flight test and airworthiness. NAVAIR verifies data obtained from aircraft manufacturers, including HV diagrams. In contrast to the major helicopter manufacturers, NAVAIR did not possess an analytical program to calculate an HV diagram. A need existed to verify manufacturers HV data not only through flight test but with analytical means. This investigation supports an effort to provide NAVAIR with means to analytically

calculate the single-engine HV diagram of a traditional helicopter.

1.5 Case Study Aircraft

The AH-1Z Cobra, shown in Fig. 1.2, was selected as the case study helicopter for this thesis due to familiarity with the aircraft and since flight test data for the AH-1Z was available from NAVAIR.

The AH-1Z is an attack helicopter manufactured by Bell Helicopter Textron, equipped with two T700-GE-401 (or 401C) engines, and a four blade, soft in-plane, bearingless, composite main rotor. The tail rotor is composed of two stiff in-plane teetering rotors with blade spacing of 90 degrees. The forward fuselage provides tandem seating for a crew of two and provides support for a target sight system sensor and a turret mount for an M197 20mm cannon. The aircraft uses skid landing gear. The tail boom is a semi-monocoque structure, mounting a tapered, cambered vertical fin, the tail rotor, and a tapered, cambered horizontal stabilizer. The horizontal stabilizer is of variable angle of incidence and is actuator controlled to reduce main rotor yoke moments. Maximum gross weight is 18,500 pounds and maximum airspeed is 200 knots calibrated airspeed. The United States Marine Corps utilizes the AH-1Z to provide offensive air support, armed escort and airborne supporting arms coordination. A detailed description of the AH-1Z may be found in [3].

The AH-1Z carries a variety of ordnance on six hardpoints mounted on a stubwing. For this thesis, the configuration considered was the clean configuration, with no ordnance mounted under the stubwing. Of note, during the design of the AH-1Z, several major changes were made, including one engine inoperable (OEI) engine rating and optimal main rotor speed. The flight test data available for the AH-1Z was published for an early prototype using the older engine rating and rotor speed. Therefore, aircraft input parameters in this thesis were matched to those used in

Figure 1.2. The Bell Textron AH-1Z Cobra

available test reports, and the older engine power rating and main rotor speed were utilized.

1.6 Research Objectives

Analytical means were sought to accurately calculate the HV diagram of a traditional, single rotor helicopter in a single-engine failure scenario. The following elements supported this overall objective:

- Develop a dynamic model of a helicopter
- Utilize pseudo-spectral optimal control software to provide optimal state and con-

trol trajectories for the descending helicopter

- Incorporate effects of pilot technique during single-engine failure events.
- Develop a solution procedure which can be easily adapted to different aircraft.
- Determine the entire HV diagram by solving the optimal control problem for different initial conditions.
- Examine the effects of improved induced velocity models.

To provide utility for future real-time optimization efforts, a dynamic model of the helicopter and an optimization scheme were sought to match flight test data with minimum computation time. A program using software available to NAVAIR, and capable of easy modification for different airframes was desired.

1.7 Thesis Overview

Chapter 1 provided thesis objectives, background HV diagram information, the research catalyst, and a brief description of the case study aircraft. Chapter 2 discusses previous research in the field of HV determination and an overview of dynamic optimization theory. Chapter 3 discusses research methodology used in this thesis, including the helicopter dynamic model and the method by which dynamic optimization was used to generate an HV curve. Chapter 4 presents and discusses the results of the HV diagram solution as applied to the AH-1Z. Chapter 5 draws conclusions and discusses recommendations for future research.

II. Background

2.1 Chapter Overview

This chapter examines previous research, flight test and software development which attempted to predict the Height-Velocity (HV) diagram. The analytical determination of the HV diagram was first empirically accomplished, but since the 1970s, HV diagrams have been determined using optimal control and constructing the HV diagram from optimal trajectories. Early HV research focused on a complete power loss and the HV diagram for autorotation. For a detailed description of autorotation refer to Section 8.5 of reference [4]. As multi-engine helicopters became more prevalent, research advanced to the single-engine HV diagram solution. This chapter discusses early HV flight tests, semi-empirical HV solutions and the prediction of the HV diagram using optimal control. Johnson [2] first solved a non-linear, optimal control trajectory of helicopter autorotation in 1977. Johnson's research is used as a framework to discuss subsequent research advancements.

2.2 Height Velocity Flight Tests and Empirical Prediction

2.2.1 Early HV Flight Tests

During the period 1965 to 1968, the Aircraft Development Service of the Federal Aviation Administration (FAA) completed an extensive flight test study of the helicopter HV diagram. Gross weight and density altitude were varied for three different single-engine helicopters, and the effect on the HV curves was reported by Hanley and Devore [5] in 1965. Hanley and Devore demonstrated that HV data from different, single-engine helicopters lay on a single curve, after the data was reduced to non-dimensional form. The report included a method for calculating the HV diagram, largely based on finding a critical velocity and critical height (the knee point of the

HV curve) and then spot checking the remainder of the curve with a standardized pilot technique. Since this early flight test effort, many test projects were conducted, mostly by the U.S. Army, to investigate different aircraft and the importance of different variables on the HV diagram.

2.2.2 Semi-Empirical Method

In 1968, Pegg [6] used the data from the FAA flight tests and provided a more extensively documented, semi-empirical method for determining the autorotative HV diagram. Pegg's method involved determining the low hover point, high hover point and the knee point of the HV diagram. Standardized equations were used to generate the HV curve from these three points. Pegg's semi-empirical method, though based on data from three different helicopters, included some assumptions, notably, that the critical height at the knee point was constant at 95 feet. Although more robust methods were needed to capture flight dynamics not observed during the Hanley tests, Pegg's method has been referenced for many years and notably was used by the Helicopter Dynamic Performance (HDP) program [7] as a starting guess for HV calculations. Since Pegg's research, different solution methods have proven to more accurately predict rotorcraft aeromechanics of the descending helicopter. The most prevalent solution has been accomplished using optimal control.

2.3 Optimal Control

Optimal control is founded on the calculus of variations, has roots in classical control and non-linear programming, and was made practical for the helicopter HV problem by the digital computer. The dynamic optimization procedure is to frame a problem by defining certain design variables as states and others as controls. The governing dynamic equations which relate various states and the controls are written

as differential equations. Constraints on initial, terminal and path values of the states and controls are written as algebraic equations. An expression is chosen to be minimized (or maximized). This expression is termed the performance measure or cost function. The dynamic optimization solution is the set of control and state trajectories which minimize the performance measure, while simultaneously satisfying the dynamics equations and constraints.

At the heart of optimal control theory are the first-order necessary conditions for optimality, known as the Karush-Kuhn-Tucker conditions. These can be found in any reliable resource on optimal control and are very clearly developed and discussed in texts by Kirk, [8] and Bryson, [9]. In basic terms for a function of several variables, the first order conditions state that an optimal point must lie at a location where the gradient of the function is zero. As applied to the optimization problem, an augmented objective function, or Lagrangian, is formed by appending the constraint equations to the objective function using Lagrange multipliers and the first order necessary conditions are shown in Table 2.1.

Table 2.1. First-Order Necessary Conditions for Optimality

1.)	Lagrangian stationary with respect to all design variables
2.)	Lagrangian stationary with respect to all Lagrange multipliers
3.)	Active inequality constraints must be feasible
4.)	Inequality constraint Lagrange multipliers must be non-negative
5.)	Gradients of the active constraints must be linearly independent

If this approach is extended to a dynamic optimization problem, what was an objective function (or cost function) is instead an objective functional, each optimal design variable is an optimal function, and the Lagrange multipliers are also functions usually termed co-states. Boundary conditions for the design variable and co-state functions are translated into transversality conditions. A list of tranvsersality conditions for different classes of optimization problems can be found in Table 5-1 of [8].

Solutions for the necessary conditions of a dynamic optimization problem for all but the most trivial cases are normally accomplished using numerical methods.

Betts [10] published an excellent survey of numerical optimal control methods which includes basic theory along with different numerical methods, their strengths, weaknesses and historical applications. Rao [1] published a separate survey and addresses many of the advances and areas of focus since the turn of the century. According to Betts, the two generally recognized frameworks for pursuing solutions to the optimal control problem are the indirect method and the direct method [10]. Also, since many numerical optimal control methods utilize some version of nonlinear programming (NLP), NLP is discussed before considering the different classes of numerical methods.

2.3.1 Nonlinear Programming

Nonlinear programming (or nonlinear optimization) is the numerical extension of classical Lagrangian optimization and involves the determination of an optimal set of design variables which minimize an objective function subject to a set of algebraic constraints. The problem is discretized into a series of nodes. Values for parameters and their derivatives are calculated at each node. As the number of nodes is increased, the discrete NLP equations begin to more closely approximate the first order conditions for optimality [10]. The NLP problem may be termed *dense* if a large percentage of the derivatives are non-zero, or *sparse* if a large percentage of the derivatives are zero.

An important gradient based, NLP method using Sequential Quadratic Programming (SQP) techniques was codified in the Sparse Nonlinear Optimizer (SNOPT) algorithm by Gill, Murray and Saunders [11]. SNOPT is designed for large-scale, nonlinear, sparse optimization problems and has been successfully used for the HV

problem in several instances.

2.3.2 Indirect Solutions

The calculus of variations provides the classical, or indirect, solution to the dynamic optimization problem. Lagrange multipliers (or co-states) are utilized to append the constraints to the cost function. The resulting augmented function is termed the Hamiltonian. The Hamiltonian is partially differentiated with respect to the states to obtain the adjoint or co-state equations. The control equations are obtained by differentiating with respect to the controls. The final time or final condition of the problem yields an algebraic equation, termed the transversality condition from which final values of the co-states can be determined. This resulting system of equations is coupled with the constraint equations to obtain the Euler-Lagrange equations. The end result is a two point or multi-point boundary value problem: initial conditions are available for the states, but only final values are available for the co-states. Several different numerical techniques are available to compute an indirect solution including indirect shooting [12], gradient restoration [13], and indirect collocation [14]. In some cases, analytical indirect solutions are also possible.

2.3.3 Direct Solutions

Direct methods do not require the analytical derivatives of the Hamiltonian used in the Euler-Lagrange equations. Rather, a direct optimal control method discretizes the dynamics and constraint equations, and transcribes the problem into a NLP problem [10]. A helpful diagram developed by Rao in [1] and shown in Fig. 2.1 depicts the three major subcomponents of numerical optimal control and differences in utilization by the direct and indirect methods.

Direct methods can be broken down by whether only the controls are parameter-

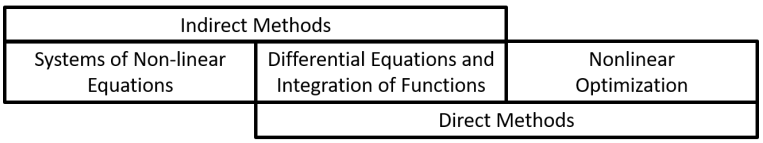


Figure 2.1. Comparison of Direct and Indirect Numerical Methods [1]

ized or whether the controls and states are parameterized. Direct shooting and direct multiple shooting are techniques in which only the controls are parameterized. In collocation methods, both the controls and states are parameterized in a collection of grid points known as a mesh. The latter method is further subdivided by local collocation and global collocation [1].

2.4 The Hamiltonian

Introduced earlier in the context of indirect methods, the continuous Hamiltonian for an optimal control problem is given by the sum of the running cost with the product of the co-state and state derivative vectors [8]. An important result of the calculus of variations is that the Hamiltonian, evaluated on an extremal trajectory, is constant for all time if the Hamiltonian is not an explicit function of time. Furthermore, if the terminal cost is not an explicit function of time, the Hamiltonian will be zero [8]. For optimal control problems where the Hamiltonian is linear with respect to the controls, part or all of the optimal trajectory lies on one or more singular arcs. Pontryagin's Minimum Principle must be added as a necessary condition for optimality in such situations, which states that the optimal control must minimize the Hamiltonian [8]. Numerical optimization methods (including direct methods) provide discrete solutions for the co-states which can be utilized to compute the Hamiltonian once an optimal solution is obtained.

2.5 The HV Diagram Determined with Optimal Control

Johnson was the first to apply optimal control theory to the non-linear helicopter dynamics equations [2]. Johnson's research was motivated by a desire to compare autorotative characteristics of different designs, and not directly concerned with finding the HV diagram, but the methods were easily adaptable for an HV diagram analysis. In this early optimal control solution of helicopter descent, many important elements were incorporated:

- Rotor stall effects
- Vortex ring state effects
- Ground effect
- First-order model of engine lag as power available decays to zero
- Delay in simulated pilot response

With the exception of rotor stall, each of these elements are examined in this thesis and remain important aspects of a dynamic optimization solution. Johnson neglected body attitudes and moments with a point mass model, and only modeled motion in the x-z plane. Dynamic equations were constructed from the force diagram shown in Fig. 2.2. Control variables were the vertical and horizontal components of the thrust coefficient. The highly non-linear relationship between rotor thrust, power and inflow was modeled using momentum theory and empirical equations which adjusted momentum theory for changes due to vortex ring state. Johnson examined a vertical descent from a hover, using the horizontal and vertical components of the thrust coefficient as controls and a quadratic cost function of vertical and horizontal final velocity. Johnson solved the resulting two point boundary value problem with an indirect solution, using an algorithm of steepest descent. Johnson observed that by assuming an "optimal" series of control inputs, pilot technique could be removed

from a comparison of different helicopter autorotative characteristics. Johnson's work established the dynamic optimization approach as a valid method to determine the optimal trajectory of a helicopter in descent.

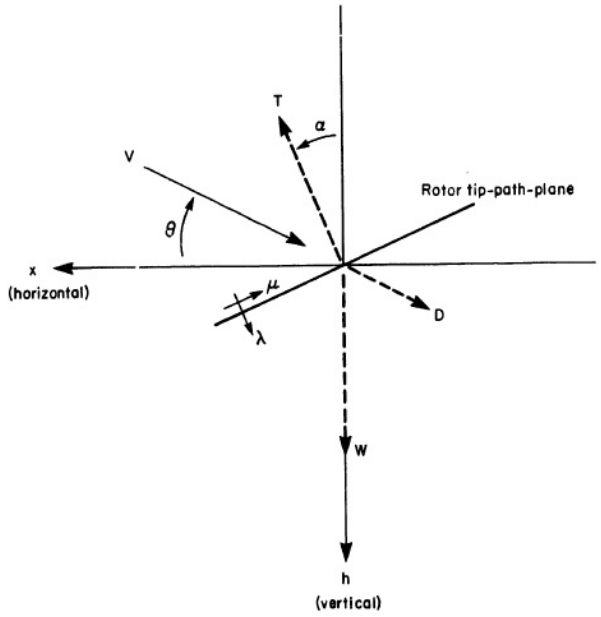


Figure 2.2. Force Diagram used by Johnson in [2]

2.6 Rotorcraft Dynamic Model Improvements

2.6.1 Rigid Body Effects and Force Balance Method

Lee [15] concluded that a point mass model was adequate for predictions of optimal descent trajectories. Okuno [16] presented a rigid body formulation, modeling fuselage pitch attitude, rotor flapping angle and motion in the x-z plane. The tip path plane angle was modeled assuming quasi-steady flapping motion. A force balance method was utilized with a modified blade element approach that accounted for the effects of blade stall. Subsequent research has demonstrated sufficient accuracy with a point-mass model and an energy balance method, seen in References [17, 18, 19, 20, 21, 22].

2.6.2 Engine Effects

With Okuno [16] as a notable exception, most formulations of the HV optimization problem use a point mass model that incorporates three degrees of freedom: vertical velocity, horizontal velocity and rotor angular velocity. To incorporate rotor angular velocity in an autorotation, Lee [15] balanced an expression for main rotor torque. Chen and Zhao [17] were the first to study a multi-engine helicopter after a single-engine failure, referred to as one engine inoperable (OEI) flight. This partial power scenario requires an expression for main rotor power vice torque. Using this energy equation, Chen and Zhao incorporated a first-order lag for the failed engine as it lost power, and to model response of the operating engine. This approach was adopted and improved by Bachelder [20] and Carlson [22] who both incorporated first-order engine control unit dynamics into their respective models.

2.6.3 Improved Model for Induced Velocity

Momentum theory diverges from reality at flight conditions where the rate of descent is approximately equal to one half the velocity induced through the rotor disk in a hover. This highly turbulent region is termed vortex ring state (VRS). In his 1977 analysis, Johnson used empirical modifications to momentum theory to eliminate the momentum theory singularity. In 2005, Johnson published a revised empirical algorithm for calculating VRS effects based on data from 17 flight, wind tunnel, whirling beam or track tests [23]. The new algorithm uses the momentum theory quartic when applicable, a baseline solution which eliminates the momentum theory singularity (similar to Johnson's previous empirical modification), and three curve fits for different segments of the VRS region which were derived from test data. The appropriate VRS region curve is chosen by the algorithm based off the ratio of forward velocity to hover inflow velocity. This new model for induced velocity

provides a close match for several different sets of test data. There is no indication that Johnson's revised algorithm for induced velocity has been implemented in recent HV research, but the algorithm holds promise for improvement of the dynamic model.

2.7 Increased Accuracy Using Path Constraints

The only constraints Johnson [2] imposed on the dynamic optimization solution were the equations of motion for a point mass helicopter. Lee [15] adopted Johnson's approach and modified it by including path constraints on the control variables and one of the state variables. Lee imposed a path inequality constraint on the thrust coefficient to limit maximum thrust to mimic a real helicopter and included an upper bound on the vertical velocity state. A solution was obtained using slack variables for the inequality constraints and the Sequential Gradient Restoration Algorithm (SGRA) developed by Miele [13]. Lee developed his solutions for the OH-58A helicopter, computed optimal descents from a hover and level flight, and was able to closely match flight test data for certain parameters.

Okuno et al. [16], used a much more complicated aerodynamic model and a rigid body model of the helicopter. His formulation included inequality constraints on blade angle of attack at 0.75 blade radius, and load factor. Okuno's formulation applied path constraints to the collective pitch, cyclic pitch, pitch attitude and rotor speed. Okuno achieved a close match to flight test data published in reference [5]. Chen and Zhao [17] used a point-mass model similar to Lee and included bounds on rotor speed, thrust and thrust angle. Chen and Zhao also departed from previous problem formulations by using the thrust coefficient time rate of change of the controls $\dot{C}_{Tx}$ and $\dot{C}_{Tz}$, to avoid discontinuity at engine failure. Bottasso [19], used this setup and applied bounds to $\dot{C}_{Tx}$ and $\dot{C}_{Tz}$. This had the effect of introducing additional dynamics into a point-mass model by modeling control system rate limits when tilting

the tip path plane. In an effort to predict the HV diagram of the AH-1Z and UH-1Y helicopters, Carlson [22] used a similar analysis but instead of bounding $\dot{C}_{Tx}$ and $\dot{C}_{Tz}$ he used $\dot{C}_T$ and $\dot{\beta}$ as controls, where β is the angle between the rotor tip path plane and the horizon. Lower and upper bounds were then applied directly to $\dot{C}_T$ and $\dot{\beta}$. The application of constraints and bounds on various states and controls has enabled the capture of complicated helicopter dynamics while retaining the simplicity of a point-mass, momentum-theory model.

2.8 Improved Numerical Solution Methods

The SGRA solution [13] originally used by Lee [24], was an indirect, numerical solution. This algorithm was also utilized by Okuno et al.[16] and Chen et al.[17] to compute optimal trajectories for a variety of flight conditions. Improvements in numerical solutions to the optimal control problem since Chen's research are described below.

2.8.1 Direct Collocation Solutions

Direct collocation, as discussed in Section 2.3.3, was first used for the descending helicopter problem by Jhemi [25] who studied optimal helicopter flight during engine failure. Research subsequent to Jhemi has mostly utilized the technique of direct collocation. Carlson [26] used SQP software developed by the Stanford Optimization Lab to optimize autorotation and single-engine failure trajectories for tilt-rotor aircraft and helicopters [26], [22]. Bottasso [19] used a finite element method of direct transcription to analyze a variety of helicopter maneuvers and to conduct parametric studies. Bachelder [20] used SQP methods and computed autorotative and OEI trajectories for comparison with trajectories. Aponso [21] utilized the same methods as Bachelder as a Real Time Optimal Control (RTOC) solution. This RTOC solution

was used to provide control cues for a simulator heads-up display, as a possible flight training aid. Direct collocation in the context of the helicopter HV problem has three major advantages over other methods. First, as opposed to any indirect technique, direct collocation does not require analytical development of the Euler-Lagrange equations, [10]. Secondly, as compared to any other method, a priori knowledge is not required of singular arcs which arise from path inequality constraints [10]. Third, as compared to direct shooting, collocation is well suited to problems involving a larger number of optimization variables [10]. Direct collocation continues to hold promise for accurate, robust solutions to the trajectory optimization of a descending helicopter.

2.9 Pseudo-Spectral Collocation and GPOPS-II®

Pseudo-spectral collocation is a particular method of direct collocation that has been heavily researched in the last two decades. The method combines global polynomial approximations for the states and controls with orthogonal collocation of the differential-algebraic equations used in a direct method. Orthogonal (or pseudo-spectral) collocation refers to the assignment of the discretization points to the roots of an orthogonal polynomial (or some combination of an orthogonal polynomial and its derivatives) [1]. The three common sets of orthogonal collocation points are the *Legendre-Gauss*, the *Legendre-Gauss-Radau*, and the *Legendre-Gauss-Lobatto* points, [1]. The *Legendre-Gauss* points include neither mesh endpoint, the *Legendre-Gauss-Radau* points include one mesh endpoint and the *Legendre-Gauss-Lobatto* points include both mesh endpoints. Furthermore, each set of points has led to a different mathematical method, namely the *Legendre Pseudo-spectral Method*, the *Radau Pseudo-spectral Method* and the *Gauss Pseudo-spectral Method*.

The University of Florida Vehicle Dynamics and Optimization Laboratory codified a Gauss pseudo-spectral method, using Gaussian quadrature to estimate the continous

cost function. Legendre polynomials were used to approximate the states and controls. Research by Liu, Hager, Darby, Patterson and Rao resulted in *hp* adaptive mesh refinement methods which increased numerical efficiency and allowed a decrease in the number of collocation points. Their adaptive methods use orthogonal collocation at the Legendre Gauss Radau (LGR) points and adapts the mesh size and/or the degree of the polynomials used to approximate the states and controls. Where the error tolerance has been met, mesh density is reduced by merging adjacent mesh intervals or by lowering the polynomial degree. These methods were combined with either an interior point NLP solver (IPOPT), or a NLP solver exploiting sparse matrix techniques using SQP methods (SNOPT). The entire software package is called the General-Purpose Pseudospectral Optimal Control Software, (GPOPS-II®) and is available commercially, [27].

GPOPS-II® was designed for multiple phase optimal control problems. Previous discussions have assumed a single phase, but multiple phases may be incorporated into a problem for a variety of reasons, the most obvious being a change in the dynamics equations [10]. If this is the case, linkage constraints must be added to the list of algebraic constraint equations ensuring that there are no discontinuities in the independent variable and the states.

Another key feature incorporated in GPOPS-II ® is the method of co-state estimation developed by Garg et al. [28]. Using an integral form of the dynamics, this method computes the integral matrix from the initial point to the interior LGR and a terminal point, as seen in Equ. (54) from [1].

2.10 H-1 Upgrades Height Velocity Diagram Development

Previously mentioned, Carlson published a method for calculating an HV diagram in [22]. This work was conducted in concert with an HV demonstration for the AH-1Z,

the results of which are available in [29]. Carlson used direct collocation techniques and the NLP solver SNOPT to predict the HV diagram prior to flight test.

The Model 449 AH-1Z Flight Test Report for Height-Velocity Demonstration, [29], contains flight test data from an HV demonstration. Section seven of this report presents a time history of a simulated single-engine failure from a low altitude, low airspeed condition. Included are values for the rotor thrust coefficient and rotor tip path plane were approximated using the method shown by Carson in [18]. This data was useful in validating the helicopter math model developed for this thesis. Section seven of [29] also contains the HV diagram which Carlson developed, along with flight test data which was subsequently obtained with an AH-1Z helicopter. The flight test data, analytically-produced HV diagram, and optimization techniques used by Carlson in [22] and [29] were extremely useful during this thesis research as a means to compare and validate results.

2.11 Conclusions from Previous Research

Recent research suggests that a momentum theory, point-mass helicopter model is adequate for HV calculations. It has been demonstrated that optimal trajectories can be used to define the boundaries of the HV diagram, and that path constraints can be used to include additional aeromechanics. The helicopter model could potentially be improved with an updated VRS model. Direct solutions using the collocation method seem to provide the most robust solution to the dynamic optimization problem and the multi-phase pseudo-spectral adaptive mesh method utilized by the GPOPS-II® algorithm holds promise as an efficient solver for the HV problem.

III. Research Methodology

3.1 Chapter Overview

This chapter examines the research methods and techniques utilized to analytically construct a Height-Velocity (HV) diagram. The case study aircraft was the Bell AH-1Z Cobra. Several flight test reports provided to NAVAIR by the AH-1Z manufacturer were available for this research. These included the AH-1Z Design Report [30], the AH-1Z Substantiation Report [31], and the H-1 Height Velocity Test Report [29]. These documents were heavily utilized for development of the methodology and comparison of results.

An incremental approach was used in this research project. The works of Johnson [2], and Lee [15], were adopted as a starting position and then modified with a series of improvements. The research by Carlson on the H-1 program documented in [29] and [22] was heavily drawn from during this thesis research. The research plan is shown below.

- Replicate Lee's optimal solution of an autorotation
- Utilize Johnson's new method to calculate induced velocity
- Add states to account for engine torque in a single-engine failure
- Add states to model limitations on control rates
- Improve accuracy of the power coefficient calculation
- Account for ground effect and vertical drag from downwash
- Adjust the helicopter model to match flight test data
- Solve the dynamic optimization problem using GPOPS-II®

The problem was coded using the Matrix Laboratory® (MATLAB) software suite and the optimization program GPOPS-II®. Unless otherwise noted, named functions refer to functions provided in MATLAB®.

3.2 Major Assumptions

The following major assumptions were used in this research:

- A three-degree of freedom, point mass model of the aircraft
- Dynamics developed from momentum theory
- Uniform rotor inflow
- Constant aircraft mass
- Constant air density
- No wind
- Ground effect model for vertical flight
- No contribution from the stub wing or the elevator

3.3 Replicating Lee's Solution

3.3.1 Aircraft Model

Using the work of Johnson [2] and Lee [15] as a guide, the equations of motion for a point-mass helicopter model were developed from the force diagram shown in Fig. 3.1.

$$m\dot{w} = mg - T\cos\alpha - D\sin\theta \tag{3.1a}$$

$$m\dot{u} = T\sin\alpha - D\cos\theta \tag{3.1b}$$

$$I_R\dot{\Omega} = -Q_{Req} \tag{3.1c}$$

The torque required was calculated using a combination of momentum theory and

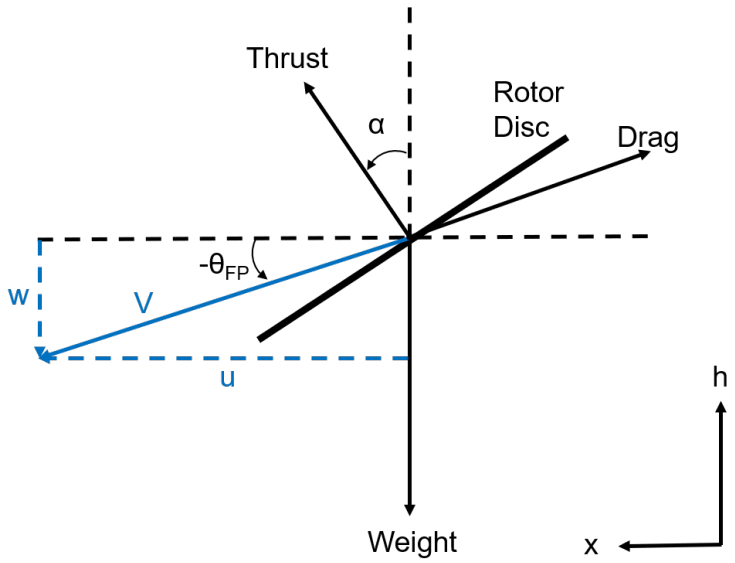


Figure 3.1. Force Diagram Used in this Research

blade element theory.

$$Q_{Req} = C_Q \rho A (\Omega R)^2 R \tag{3.2a}$$

$$C_Q = C_P \tag{3.2b}$$

$$C_P = \frac{1}{8}\sigma \bar{c}_d + C_T \lambda \tag{3.2c}$$

The rotor inflow was calculated using Johnson's technique from [2], as implemented by Lee in pages 19-20, and 28-29 of [15]. This technique used a single, empirically derived equation to account for the momentum theory singularity.

Aircraft parameters were assigned to replicate the OH-58A helicopter equipped with high inertia blades, matching the inputs used by Lee.

3.3.2 Optimization Problem Setup Using Lee's Method

A general formulation of a dynamic optimization problem is given below.

$$\begin{aligned} \underset{u}{\text{minimize}} \quad & J = \phi(x_f, u_f, t_f) + \int_{t_0}^{t_f} L(x, u, t)dt \\ \text{subject to} \quad & \mathbf{x}' - \mathbf{f} = 0, \\ & \Psi(x_f, u_f, t_f) = 0 \end{aligned} \tag{3.3}$$

where:

J = Performance Measure	x = States
ϕ = Terminal Cost	u = Controls
L = Path Cost	t = Time
f = Path Constraints	
Ψ = Terminal Constraints	

States were defined and transformed into non-dimensional form to match Lee's problem setup on pages 26-34 of [15]. State and control definitions are included here for clarity as Lee's formulation was the starting point for research undertaken in this research.

$$x_1 = \frac{w}{0.01\Omega_0 R} \tag{3.4a}$$

$$x_2 = \frac{u}{0.01\Omega_0 R} \tag{3.4b}$$

$$x_3 = \frac{\Omega}{\Omega_0} \tag{3.4c}$$

$$x_4 = \frac{h}{10R} \tag{3.4d}$$

$$x_5 = \frac{x}{10R} \tag{3.4e}$$

$$u_1 = 1000 C_{Tx} \tag{3.5a}$$

$$u_2 = 1000 C_{Tz} \tag{3.5b}$$

Time was non-dimensionalized using the reference time shown in Equ 3.6a and 3.6b. The derivative with respect to non-dimensional time τ is depicted in Equ. 3.6d. Prime notation ($'$) indicates the derivative with respect to non-dimensional time, τ.

$$\tau = \frac{t}{t_{ref}} \tag{3.6a}$$

$$t_{ref} = \frac{100}{\Omega_0} \tag{3.6b}$$

$$\tau = \frac{\Omega_0 t}{100} \tag{3.6c}$$

$$\frac{d}{d\tau} = \frac{100}{\Omega_0}\frac{d}{dt} \tag{3.6d}$$

Using the previous definition of prime notation and the constants defined on page 27 of [15], the following equations of motion were used to define the dynamic optimization problem dynamic and kinematic relationships.

$$x_1' = g_0 - m_0 \left(u_1 x_3^2 + \bar{f} x_1 \sqrt{x_1^2 + x_2^2} \right) \tag{3.7a}$$

$$x_2' = m_0 \left(u_2 x_3^2 + \bar{f} x_2 \sqrt{x_1^2 + x_2^2} \right) \tag{3.7b}$$

$$x_3' = -i_0 x_3^2 \left(c_0 + \lambda \sqrt{u_1^2 + u_2^2} \right) \tag{3.7c}$$

$$x_4' = 0.1 x_1 \tag{3.7d}$$

$$x_5' = 0.1 x_2 \tag{3.7e}$$

Initial conditions for the kinematic states were determined by the initial flight condition. Initial values for the remaining states and controls were found by simultaneously solving a system of four equations. The *fsolve* function was used to solve for the initial conditions.

3.3.3 Solution Using *fmincon*

MATLAB® script was written using the *fmincon* function. The *fmincon* algorithm solves a dynamic optimization problem by formulating the constraint equations at discrete time steps and solving the resulting system of equations as a static optimization problem.

The constraint and objective functions used by Lee were written in discrete time, using a reference time of approximately 2.7 seconds. Constrained final state, free final time solution cases were run from a 100 ft hover and from 100 ft and 12 knots airspeed, with a constraint on maximum C_T. For the hover case, the optimal control trajectory was obtained with the Sequential Quadratic Programming (SQP) solver provided in *fmincon*. For the 100 foot, 12 knot case, the SQP algorithm did not converge so the interior-point algorithm was utilized. The constraint and convergence tolerances were both set at $1\text{x}10^{-6}$. Results showed good agreement to Lee's solution found in Chapter 4 of [15].

The optimal control and state histories obtained with *fmincon* showed very similar trends as Lee's solutions. For both cases, the *fmincon* solution found slightly different optimal control strategies and final times, but these are attributed to the different solution method of *fmincon* vice the Sequential Gradient Restoration Algorithm (SGRA). The *fmincon* solution of the 100 foot 12 knot point shows some irregularities and significant differences, but again trends are matched to Lee's solutions.

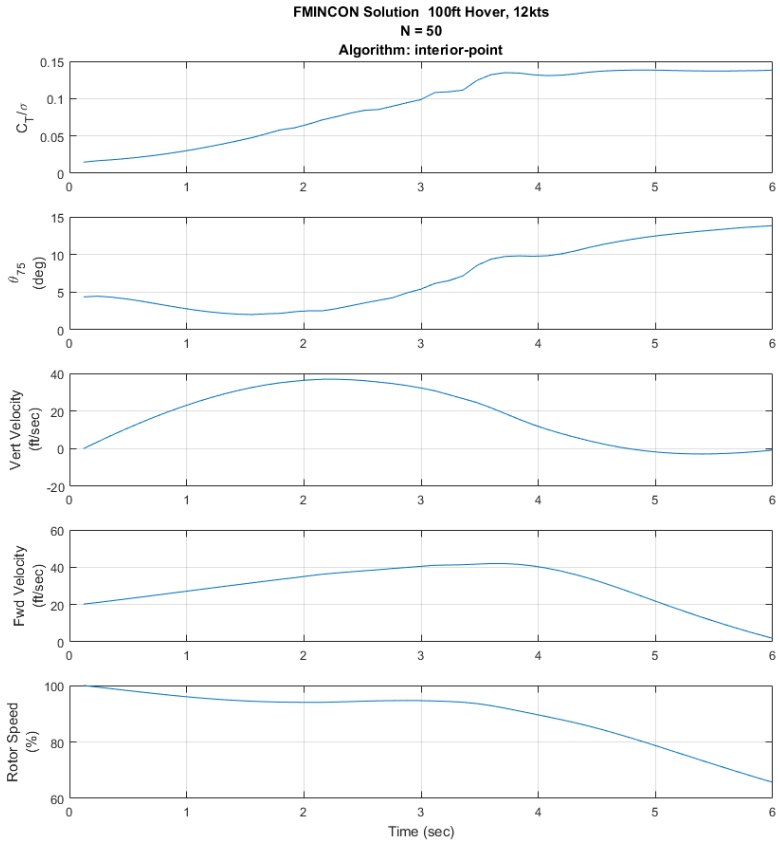


Figure 3.2. Solution of Lee's Problem with *fmincon*

The solution obtained with *fmincon* was extremely sensitive to the number of discrete intervals used to solve the static optimization problem. As can be seen in Fig. 3.3, the angle of the tip path plane to the horizon is quite extreme and requires additional constraints. Due to the method by which variables were passed to the *fmincon* function, additional constraints were not feasible. A better solution technique was sought.

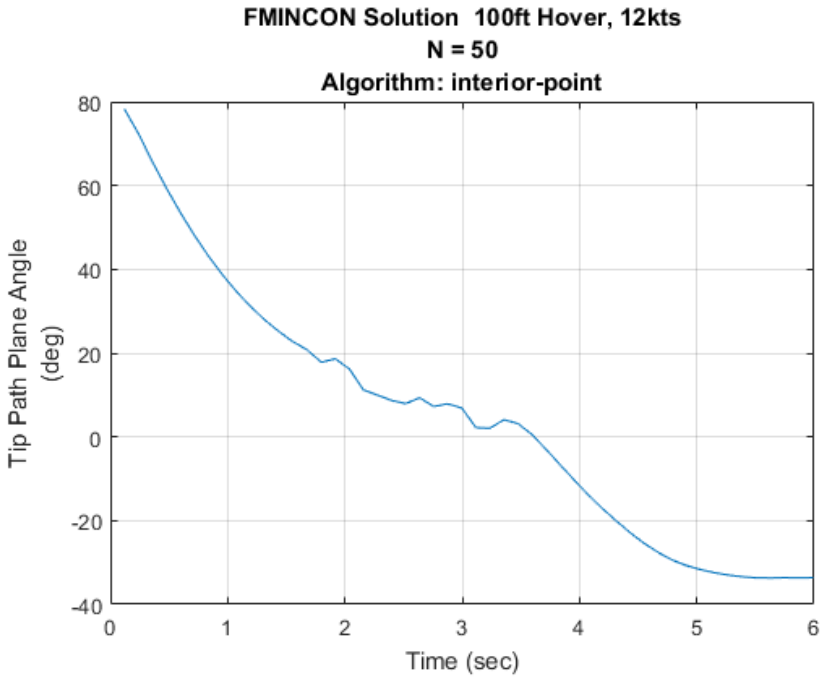


Figure 3.3. Un-Constrained Rotor Tip Path Plane Angle

3.4 Changes to the Helicopter Model

Using the math model as derived by Johnson and Lee as a starting point, several additions and improvements were incorporated. First, the problem was formulated using the time rate of change of rotor tip path plane angle and the time rate of change of the thrust coefficient as the dynamic optimization controls. Four additional states were added to utilize the accelerations of thrust and rotor tip path plane angle as the controls. Second, an improved method was incorporated to calculate the rotor inflow. Third, two states were added to account for engine power available during a single-engine failure on a twin engine helicopter. Fourth, the method for calculating the power coefficient was modified to improve accuracy over a broader span of flight conditions. Finally, factors were introduced to the dynamics equations to incorporate ground effect and fuselage vertical drag from rotor downwash. The methodology for these improvements was adopted from references [16, 17, 18, 20, 21, 23] and especially

[22]. In the following paragraphs, each modification is addressed in greater detail. Of note, state definitions for states x_1 through x_5 were unchanged, and the definition of τ was unchanged.

3.4.1 Control Definitions and Associated Additional States

The dynamics equations were written to utilize the time rate of change of the normalized thrust coefficient and the time rate of change of the rotor tip path plane angle as the two controls. Two additional states were added to integrate the controls for inclusion in the equations of motion. This allowed constraints to be placed on the thrust coefficient rate of change and the tip path plane angle rate of change. These constraints were used to mimic un-modeled dynamics of the rotor system. The additional states of the integrated controls (x_8 and x_9) which represented the thrust coefficient and tip path plane angle were constrained to mimic rotor performance and pitch attitude limitations.

$$u_1 = 1000\dot{C}_T \tag{3.8a}$$

$$u_2 = \dot{\alpha} \tag{3.8b}$$

$$x_8 = 1000C_T \tag{3.8c}$$

$$x_9 = \alpha \tag{3.8d}$$

During further progression in the research process, two additional states were added and were integrated in a similar manner. The accelerations of thrust coefficient

and rotor tip path plane angle were then used as controls.

$$u_1 = 1000\ddot{C}_T \tag{3.9a}$$

$$u_2 = \ddot{\alpha} \tag{3.9b}$$

$$x_{10} = 1000\dot{C}_T \tag{3.9c}$$

$$x_{11} = \dot{\alpha} \tag{3.9d}$$

3.4.2 Modifications for More Accurate Rotor Inflow

The rotor inflow was calculated using an algorithm developed by Johnson [23]. The components of velocity were calculated perpendicular and parallel to the rotor tip path plane and were made non-dimensional using the rotor inflow in a hover, v_h.

$$v_h = \sqrt{\frac{T}{2\rho A}} \tag{3.10a}$$

$$\mu_z = \left(\frac{\text{Rotor } V_z}{v_h}\right) = \frac{u\sin\alpha - w\cos\alpha}{v_h} \tag{3.10b}$$

$$\mu_x = \left(\frac{\text{Rotor } V_x}{v_h}\right) = \frac{u\cos\alpha + w\sin\alpha}{v_h} \tag{3.10c}$$

Using the previous state definitions, the following equations were obtained.

$$v_h = R\Omega_0 x_3\sqrt{\frac{x_8}{2000}} \tag{3.11a}$$

$$\mu_z = \left(\frac{0.4472}{x_3\sqrt{x_8}}\right)(x_2\sin x_9 - x_1\cos x_9) \tag{3.11b}$$

$$\mu_x = \left(\frac{0.4472}{x_3\sqrt{x_8}}\right)(x_2\cos x_9 + x_1\sin x_9) \tag{3.11c}$$

These non-dimensional velocities were passed to a series of functions which re-

turned a value for the non-dimensional, induced rotor inflow, λ_{iJ}, using the method detailed in Chapter 2. Since this value was made non-dimensional with v_h, the parameter was converted into the traditional λ_i by the following equation:

$$\lambda_i = \frac{\lambda_{iJ} v_h}{\Omega R} \tag{3.12}$$

In terms of states this equation was written as follows.

$$\lambda_i = 0.0224 \lambda_{iJ} \sqrt{x_8} \tag{3.13}$$

For flight conditions in which the momentum theory result was valid, the momentum theory quartic was solved using a Newton-Raphson technique developed by Johnson in Chapter 5 of [4]. Since the rotor inflow was calculated using momentum theory for much of the flight regime, this solution technique drastically reduced computation time, as compared to solutions using *fzero* to solve the momentum quartic.

3.4.3 Additional States for Engine Power

Following the works of Bachelder [20] and Carlson [22], two states were added to account for an operating engine during single-engine failure. The first state represented the failed engine, and provided power that exponentially decayed to zero. The second state was modeled with a proportional controller, and included saturation at the one engine inoperable (OEI) power rating, the maximum power output of the engine as governed by an engine controller. Specific to the AH-1Z, a half second delay was included to model engine control unit response to a single-engine failure. Each

power state was normalized with the OEI power rating.

$$\dot{P}_1 = -\frac{1}{\tau_1} P_1 \tag{3.14a}$$

$$\dot{P}_2 = 0 \text{ for t} \leq 0.5 \tag{3.14b}$$

$$\dot{P}_2 = \frac{1}{\tau_2}(P_{2AG} - P_2) \text{ for t} > 0.5 \tag{3.14c}$$

$$P_{2AG} = min\Big\{P_R - G(\Omega - \Omega_0), P_{OEI}\Big\} \tag{3.14d}$$

Each of the above equations was divided by P_{OEI} yielding the following non-dimensional governed engine power available and power state derivatives.

$$x_6' = -\frac{100}{\Omega_0 \tau_1} x_6 \tag{3.15a}$$

$$x_7' = 0 \text{ for } \tau \leq \frac{0.5}{t_{ref}} \tag{3.15b}$$

$$x_7' = \frac{100}{\Omega_0 \tau_1}(P_{2G} - x_7) \text{ for } \tau > \frac{0.5}{t_{ref}} \tag{3.15c}$$

$$P_{2G} = min\Big\{P_{R1} - \frac{G\Omega_0}{P_{OEI}}(x_3 - 1), 1\Big\} \tag{3.15d}$$

3.4.4 Power Coefficient and Total Power Required

The power required was calculated as a sum of the power required for the main rotor, tail rotor and accessory/drivetrain loss. The formulation for accessory and drivetrain losses is specific to the drivetrain design of the AH-1Z. The remaining terms were formulated generically for a traditional helicopter.

3.4.4.1 Main Rotor Power

The main rotor power coefficient used by Johnson [2], Lee [24] and Aponso [21] was modified to increase accuracy over a larger range of airspeeds. The power coefficient

for the main rotor represents all sources of rotor energy loss, and was calculated as the sum of the power coefficients for induced power, profile power, parasite power and climb power respectively. Each term was calculated using techniques from Chapter 6 of [4]. The equations were solved using the MATLAB® *fsolve* function to determine power required in several level flight conditions. The resulting values were compared to level flight performance for the AH-1Z provided by NAVAIR, and adjustments to the model were made to match flight test data as closely as possible.

$$C_{P_{MR}} = C_{Pi} + C_{Po} + C_{Pp} + C_{Pc} \tag{3.16}$$

The induced power coefficient, C_{pi}, was calculated assuming uniform inflow. An induced velocity correction factor, k_i was applied.

$$C_{Pi} = k_i \lambda_i C_T \tag{3.17}$$

The profile power coefficient, C_{Po}, was augmented with a $(1-4.65\mu^2)$ term. According to Chapter 6 of [4], this approximation yields less than 1% error for μ ranging from 0 to 0.35. For the AH-1Z case study helicopter, a μ of 0.35 corresponds to approximately 149 knots, well outside the single-engine flight envelope.

$$C_{Po} = \frac{\sigma c_{do}}{8}(1 + 4.65\mu^2) \tag{3.18}$$

With μ, the advance ratio, calculated as follows in terms of dimensional and non-dimensional parameters, respectively.

$$\mu = \frac{u\cos\alpha + w\sin\alpha}{\Omega R} \tag{3.19a}$$

$$\mu = \frac{0.01}{x_3}(x_2\cos x_9 + x_1 \sin x_9) \tag{3.19b}$$

The parasite power coefficient was calculated using the following approximation from Chapter 5 of reference [4].

$$C_{Pp} = \frac{1}{2}\frac{f_{ex}}{A}\mu^3 \tag{3.20}$$

The climb power coefficient accounts for the energy required to change altitude. Climb power was calculated as the product of aircraft weight and vertical velocity, and a negative sign was required since w was defined positive down in the problem setup.

$$P_c = -wW \tag{3.21}$$

$$C_{Pc} = -\frac{wW}{\rho A(\Omega R)^3} \tag{3.22}$$

3.4.4.2 Tail Rotor Power

In lieu of a similar buildup for the tail rotor power coefficient, $C_{P_{TR}}$, $C_{P_{TR}}$ was calculated using a power coefficient formulated for hover. The tail rotor thrust required and thrust coefficient were calculated using main rotor torque. An approximation for the tail rotor power coefficient was then calculated using a power coefficient for a hovering rotor, and modified with a dependence on advance ratio which was empirically derived in this research. Due to the relatively small power required for the tail rotor compared to the main rotor, the incurred error was judged to have negligible impact on the solution.

$$T_{TR} = \frac{Q_{MR}}{l_{TR}} \tag{3.23a}$$

$$C_{T_{TR}} = \frac{T_{TR}}{\rho A_{TR}(\Omega_{TR}R_{TR})^2} \tag{3.23b}$$

$$P_{R_{TR}} = (1 - 1.2\mu)\frac{1}{M_{TR}}C_{T_{TR}} \tag{3.23c}$$

3.4.4.3 Drive Train Losses and Accessory Power

Using the method and values for the efficiencies shown in Chapter 8 of [31], the drive train losses of the main rotor, tail rotor and combining gearbox were considered separately for the AH-1Z. The method used to calculate drivetrain efficiencies will vary from one aircraft to another and is dependent on the drivetrain design.

3.4.4.4 Total Power Required

The total power required was calculated using the method shown on page 8-1 of [31]. The power required was made non-dimensional by dividing by the OEI power.

$$P_R = \frac{1}{\eta_{CBOX}} \left(\frac{P_{MR}}{\eta_{MR}} + \frac{P_{TR}}{\eta_{TR}} + P_{Acc} \right) \tag{3.24a}$$

$$P_{R1} = \frac{P_R}{P_{OEI}} \tag{3.24b}$$

3.4.5 Ground Effect

Ground effect was incorporated in the model using an approach found in [4], [32], [33] and [34]. Ground effect may be treated by modifying the induced velocity experienced by the main rotor at a constant thrust or by modifying the thrust which the rotor produces at a constant induced power. The latter approach was used in [4] and was adopted for this investigation. A ground effect factor, k_G was defined as the ratio of thrust produced in ground effect, to thrust produced out of ground effect.

$$k_G = \frac{T_{IGE}}{T_{OGE}} \tag{3.25}$$

Chapter 4 of reference [4] includes several different methods to calculate k_G, each a function empirically derived from test data. For all methods, k_G is a function of

the ratio of rotor altitude (z) to the rotor radius. Above approximately two rotor diameters, k_G is unity. As altitude decreases, k_G increases to a maximum value of approximately 1.31. In the equations of motion, k_G was used as a multiplicative factor on C_T in the dynamic equations only.

Data was available in [31] for values of $\frac{1}{k_G}$ for the AH-1Z which were derived from flight test. The Cheeseman/Bennett solution and the Hayden solution were compared to the data in [31] and the Hayden ground effect solution was found to be a very close match. The Hayden solution was used to run simulations. Since the equation returns values slightly greater than one for all altitudes, k_G was set to 1.0 when the altitude was greater than twice the rotor diameter. Fig 3.4 shows a comparison of the Hayden and Cheeseman/Bennett solutions.

Cheeseman and Bennett ground effect factor:

$$k_G = \left[1 - \frac{1}{(4z/R)^2}\right]^{-1} \tag{3.26}$$

Hayden ground effect factor:

$$k_G = \left[0.9926 + \frac{0.03794}{(z/2R)^2}\right]^{2/3} \tag{3.27}$$

3.4.6 Vertical Drag from Downwash

The rotor wake impinges on a helicopter fuselage and creates a force in the z aircraft body axis. According to Johnson in [4], this force can be treated as a modification of the aircraft thrust. Johnson also states that vertical drag from downwash is airframe and rotor system dependent, that ground effect may cause an unpredictable change in the vertical drag force (including a different direction of the force) and that the vertical drag force disappears above transition velocity in forward flight. For this

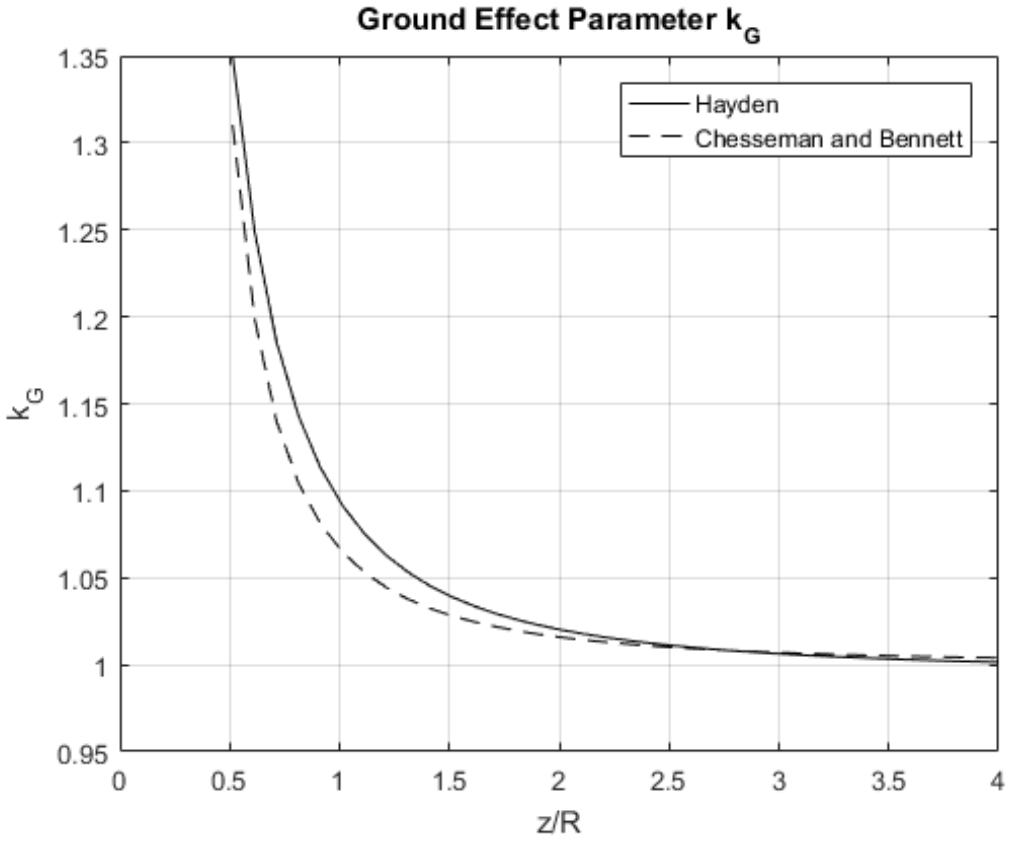


Figure 3.4. Ground Effect Models

research, the factor f_v was used to modify the thrust coefficient in the helicopter math model.

$$f_v = \frac{\Delta T}{T} = \frac{f_{eR}}{A - A_{cuff}} \tag{3.28}$$

A washout parameter, f_w, was included to reduce the vertical drag force as airspeed increased. The value of f_w in a hover was one, and linearly decreased to zero at an airspeed of 30 knots.

3.4.7 Pilot Delay

HV diagrams are typically computed or tested with the incorporation of artificial delay to simulate the reaction time of an operational pilot. A pilot delay was modeled by integrating the dynamics equations using the *ode45* function. Prior to the inte-

gration, trim values were calculated for the engine power states and controls, using the same method as described in Section 4.2.1. The computed values for the thrust coefficient and rotor tip path plane angle were held constant during the integration. A 1.5 second delay was used to determine the portion of the HV diagram above the knee point and zero delay was used below the knee point. State values at the end of the one second integration were used as initial conditions for the optimization problem.

3.5 Dynamic Model

The dimensional dynamic equations for the helicopter model previously described are shown below. These seven equations were formulated into two different models in this research. The first was a nine-state, two control model similar to that of Carlson, [29], which used the first derivatives of thrust coefficient and tip path plane angle as the two controls. The second formulation was an 11 state model, using the second derivatives of thrust coefficient and rotor tip path plane angle as the controls. The nine-state model is shown below with definitions of states, controls, dynamic equations, supporting parameters and constants. Modifications for the 11 state model are also depicted.

3.5.1 Dimensional Equations

$$m\dot{w} = mg - T(k_g + f_v f_w)\cos\alpha - D\sin\theta \tag{3.29a}$$

$$m\dot{u} = T(k_g + f_v f_w)\sin\alpha - D\cos\theta \tag{3.29b}$$

$$I_R\dot{\Omega}\Omega = P_1 + P_2 - P_{Req} \tag{3.29c}$$

$$\dot{h} = -w \tag{3.29d}$$

$$\dot{x} = u \tag{3.29e}$$

$$\tau_1\dot{P}_{S1} = -P_{S1} \tag{3.29f}$$

$$\tau_2\dot{P}_{S2} = P_{A2G} - P_{S2} \tag{3.29g}$$

3.5.2 State Definitions

$$x_1 = \frac{w}{0.01\Omega_0 R} \tag{3.30a}$$

$$x_2 = \frac{u}{0.01\Omega_0 R} \tag{3.30b}$$

$$x_3 = \frac{\Omega}{\Omega_0} \tag{3.30c}$$

$$x_4 = \frac{h}{10R} \tag{3.30d}$$

$$x_5 = \frac{x}{10R} \tag{3.30e}$$

$$x_6 = \frac{P_1}{P_{OEI}} \tag{3.30f}$$

$$x_7 = \frac{P_2}{P_{OEI}} \tag{3.30g}$$

$$x_8 = 1000C_T \tag{3.30h}$$

$$x_9 = \alpha \tag{3.30i}$$

For the 11 state model:

$$x_{10} = 1000\dot{C}_T \tag{3.31a}$$

$$x_{11} = \dot{\alpha} \tag{3.31b}$$

$$\tag{3.31c}$$

3.5.3 Control definitions

$$u_1 = 1000\dot{C}_T \tag{3.32a}$$

$$u_2 = \dot{\alpha} \tag{3.32b}$$

For the 11 state model:

$$u_1 = 1000\ddot{C}_T \tag{3.33a}$$

$$u_2 = \ddot{\alpha} \tag{3.33b}$$

3.5.4 Dynamic Equations

The system of dynamics equations, $\mathbf{f}$, is defined as

$$\mathbf{f} = \mathbf{x}' \tag{3.34}$$

$\mathbf{x}'$ is composed of the individual equations of motion, and the $'$ notation denotes the derivative with respect to non-dimensional time τ, as given in Eq. 3.6d.

$$x_1' = g_0 - m_0 x_3^2 x_8 (k_g + f_w f_v) \cos x_9 - f_{0Z} x_1 \sqrt{x_1^2 + x_2^2} \tag{3.35a}$$

$$x_2' = m_0 x_3^2 x_8 (k_g + f_w f_v) \sin x_9 - f_{0X} x_2 \sqrt{x_1^2 + x_2^2} \tag{3.35b}$$

$$x_3' = \frac{i_0}{x_3}(x_6 + x_7 - P_{R1}) \tag{3.35c}$$

$$x_4' = -0.1 x_1 \tag{3.35d}$$

$$x_5' = 0.1 x_2 \tag{3.35e}$$

$$x_6' = -k_1 x_6 \tag{3.35f}$$

$$x_7' = k_2(P_{2G} - x_7) \tag{3.35g}$$

$$x_8' = t_{ref} u_1 \tag{3.35h}$$

$$x_9' = t_{ref} u_2 \tag{3.35i}$$

For the 11 state model:

$$x_8' = t_{ref} x_{10} \tag{3.36a}$$

$$x_9' = t_{ref} x_{11} \tag{3.36b}$$

$$x_{10}' = t_{ref} u_1 \tag{3.36c}$$

$$x_{11}' = t_{ref} u_2 \tag{3.36d}$$

3.5.5 Supporting Parameters

The following expressions provide the main rotor and tail rotor power equations and the total power required in dimensional and non-dimensional terms.

$$C_{P_{MR}} = k_i \lambda_i \frac{u_1}{1000} + p_1(1 + 4.65\mu^2) + p_2\mu^3 - p_3 \frac{x_1}{x_3^3} \tag{3.37a}$$

$$P_{R_{MR}} = \frac{1}{M} p_4 C_{P_{MR}} x_3^3 \tag{3.37b}$$

$$P_{R_{TR}} = (1 - 1.2\mu) \frac{1}{M_{TR}} \sqrt{\frac{P_{R_{MR}}^3}{p_T x_3^3}} \tag{3.37c}$$

$$P_R = \frac{1}{\eta_{cbox}} \left(\frac{P_{MR}}{\eta_{MR}} + \frac{P_{TR}}{\eta_{TR}} + P_{Acc} \right) \tag{3.37d}$$

$$P_{R1} = \frac{P_R}{P_{OEI}} \tag{3.37e}$$

$$P_{2G} = min\Big\{ P_{R1} - G_1(x_3 - 1), 1 \Big\} \tag{3.37f}$$

3.5.6 Constants

Constants used in the preceding equations are defined below.

$$
\begin{aligned}
f_{0x} &= \frac{\rho f_{ex} R}{2m} \\
f_{0z} &= \frac{\rho f_{ez} R}{2m} \\
g_0 &= \frac{10000 g}{\Omega_0^2 R} \\
i_0 &= \frac{100 P_{OEI}}{I_R \Omega_0^3} \\
k_1 &= \frac{t_{ref}}{\tau_1} \\
k_2 &= \frac{t_{ref}}{\tau_2} \\
G_1 &= \frac{G\Omega_0}{P_{OEI}} \\
m_0 &= \frac{10\rho\pi R^3}{m} \\
p_1 &= \frac{\sigma c_{d0}}{8} \\
p_2 &= \frac{f_{ex} + \delta f_e}{2\pi R^2} \\
p_3 &= \frac{0.01W}{\rho\pi R^4 \Omega_0^2} \\
p_4 &= \rho\pi R^5 \Omega_0^3
\end{aligned}
$$

3.6 Optimization Problem Setup for GPOPS-II®

The helicopter models developed in preceding paragraphs were used to formulate optimal control problems useful for determining the HV diagram. Although several different permutations of the problem were utilized, the following problem setup addresses the final, 11 state model.

The problem was framed separately for the portion of the HV curve above the

knee and the portion below the knee. Above the knee, the problem was solved in two phases with an objective function based on initial altitude and minimum control. The two phases allowed the incorporation of more stringent pitch attitude limits near the deck. Since the helicopter was modeled as a point mass, rotor angle was used to approximate pitch attitude. This approximation was used by Carlson in [29] to derive values for rotor angle from pitch attitude data. The boundary between the two phases was set to occur as the helicopter model descended below three feet. For the second phase, with altitude less than three feet, path bounds on rotor tip path plane angle were reduced to $\pm$ 5deg, and the final value of rotor angle was bounded with values similar to actual aircraft touchdown attitude.

Below the knee a single phase was used. A single phase was sufficient since flight test data showed that the more stringent bounds used below three feet as described above were very close to those used throughout the trajectory for lower curve data points. Bounds on the rotor tip path plane angle were set to the same values as in phase two described above.

Both formulations included an open final time, a partially constrained initial state and partially constrained final state. The dynamics equations were enforced as path constraints and bounds were placed on all states and controls.

3.6.1 Optimal Control Problem Above the Knee Point

$$\begin{aligned}
\underset{u}{\text{minimize}} \quad & J = h_0 + \int_{\tau_0}^{\tau_f} W_1 u_1^2 + W_2 u_2^2 \, dt \\
\text{subject to} \quad & \mathbf{x}' - \mathbf{f} = 0, \\
& x(\tau_0) - x_{postdelay} = 0, \\
& g_1(x) \leq 0, \\
& g_2(u) \leq 0, \\
& x_f^{p1} - x_0^{p2} = 0, \\
& x(\tau_f) - x_f = 0
\end{aligned} \tag{3.39}$$

Note that the initial altitude state (x_4) was free and that only certain final states were fixed with equality constraints. Functions g_1 and g_2 refer to the path bounds on the states and controls respectively.

3.6.2 Optimal Control Problem Below the Knee Point

$$\begin{aligned}
\underset{u}{\text{minimize}} \quad & J = -h_0 \\
\text{subject to} \quad & \mathbf{x}' - \mathbf{f} = 0, \\
& x(\tau_0) - x_{postdelay} = 0, \\
& g_1(x) \leq 0, \\
& g_2(u) \leq 0, \\
& x(\tau_f) - x_f = 0
\end{aligned} \tag{3.40}$$

The major differences from the previous setup are the objective function and lack of phase linkage constraints since a single phase was utilized below the knee. Again the initial altitude state (x_4) was free and only certain final states were fixed.

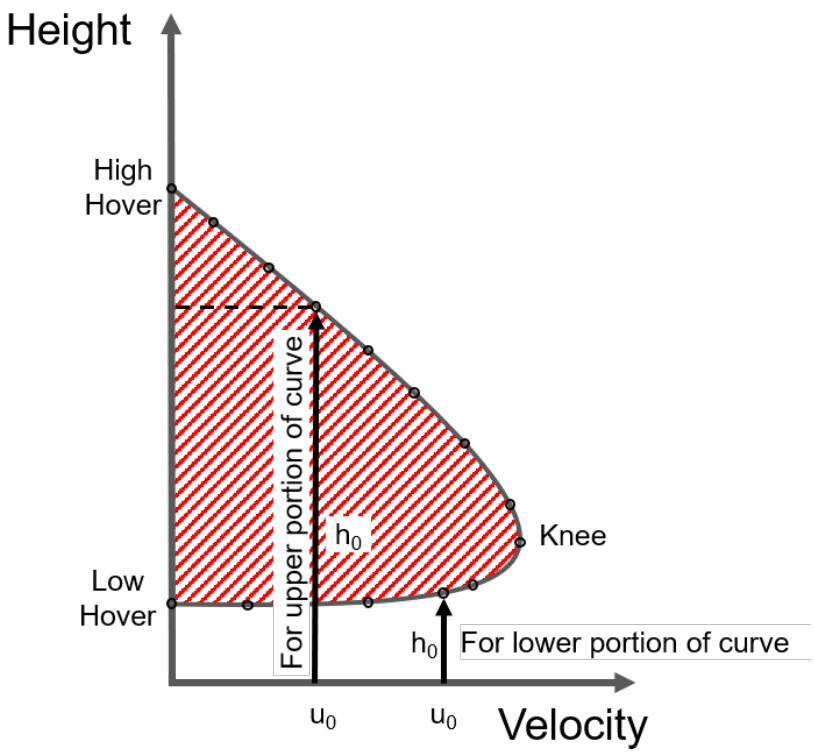


Figure 3.5. Objective Function Definition

3.6.3 Initial Condition Constraints

After the integration of the dynamics equations to simulate pilot delay, end values for each state were enforced as initial conditions for the optimal control problem. Initial altitude remained free. The initial conditions for the controls were set to zero. The constraints listed below are given in terms of dimensional variables and were

converted to non-dimensional states or controls prior to utilization.

$$
\begin{aligned}
t_0 &= t_{delay} \\
w_0 &= \text{Vertical velocity after pilot delay} \\
u_0 &= \text{given value for each iteration} \\
\Omega_0 &= \text{rotor angular velocity after pilot delay} \\
x_0 &= 0 \\
P_{1_0} &= \text{Engine 1 power after pilot delay} \\
P_{2_0} &= \text{Engine 2 power after pilot delay} \\
C_{T_0} &= \text{trim condition } C_T \\
\alpha_0 &= \text{trim condition } \alpha \\
\dot{C}_{T_0} &= 0 \\
\dot{\alpha}_0 &= 0 \\
\ddot{C}_{T_0} &= 0 \\
\ddot{\alpha}_0 &= 0
\end{aligned}
$$

3.6.4 Path Constraints

The following inequalities were enforced as path constraints. Airspeed was limited to 150 knots, and sink rate was limited to 3000 feet per minute. The choicees of α_{min}

and α_{max} had a great effect on the solution and are discussed in Chapter 4.

$$
\begin{aligned}
t_{delay} \leq\ & t & \leq t_{max} \\
0 \leq\ & w & \leq w_{max} \\
0 \leq\ & u & \leq u_{max} \\
\Omega_{min} \leq\ & \Omega & \leq \Omega_{max} \\
0 \leq\ & h & \leq h_0 \\
0 \leq\ & P_1 & \leq P_{OEI} \\
0 \leq\ & P_2 & \leq P_{OEI} \\
C_{Tmin} \leq\ & C_T & \leq C_{Tmax} \\
\alpha_{min} \leq\ & \alpha & \leq \alpha_{max} \\
\dot{C}_{Tmin} \leq\ & \dot{C}_T & \leq \dot{C}_{Tmin} \\
\dot{\alpha}_{min} \leq\ & \dot{\alpha} & \leq \dot{\alpha}_{max} \\
\ddot{C}_{Tmin} \leq\ & \ddot{C}_T & \leq \ddot{C}_{Tmax} \\
\ddot{\alpha}_{min} \leq\ & \ddot{\alpha} & \leq \ddot{\alpha}_{max}
\end{aligned}
$$

3.6.5 Linkage Constraints

For the upper portion of the curve, two phases were utilized. Linkage constraints were used to knit the time and state values together at the border of the first and second phase, ensuring continuity of the control, state and time solution from phase one to phase two.

$$\text{for k} = 1 \text{ to } 9,\ x_{k_f}^{p1} = x_{k_0}^{p2}$$

3.6.6 Final Condition Constraints

The final altitude was fixed to zero with an equality constraint, ensuring the solution terminated at the ground. Tighter bounds were placed on vertical velocity,

horizontal velocity and rotor tip path plane angle to ensure a safe touchdown.

$$h_f = 0$$

$$\begin{aligned}
w_{fmin} \leq\ & w_f \leq w_{fmax} \\
u_{fmin} \leq\ & u_f \leq u_{fmax} \\
\Omega_{min} \leq\ & \Omega_f \leq \Omega_{max} \\
0 \leq\ & P_1 \leq P_{OEI} \\
0 \leq\ & P_2 \leq P_{OEI} \\
C_{Tmin} \leq\ & C_T \leq C_{Tmin} \\
0 \leq\ & \alpha \leq \alpha_f \\
\dot{C}_{Tmin} \leq\ & \dot{C}_T \leq \dot{C}_{Tmin} \\
\dot{\alpha}_{min} \leq\ & \dot{\alpha} \leq \dot{\alpha}_{max} \\
\ddot{C}_{Tmin} \leq\ & \ddot{C}_T \leq \ddot{C}_{Tmin} \\
\ddot{\alpha}_{min} \leq\ & \ddot{\alpha} \leq \ddot{\alpha}_{max}
\end{aligned}$$

3.7 Solution using GPOPS-II®

3.7.1 Solution-Specific Input Parameters

There were several parameters which were varied for different data points on the HV curve. The most important of these by far was the choice of state and control bounds, but each was noted to have an affect on the solution.

Table 3.1. Parameters Varied for Each Run

Bounds on tip path plane angle
Bounds on tip path plane rate
Guess for the initial altitude
Upper and lower bounds on initial altitude

3.7.2 Static Input Parameters

Other parameters in GPOPS-II® were set by trial and error, and remained mostly unchanged for all computations. Values are listed in Table 3.2 with explanations in the paragraphs below, if required. Unless otherwise noted, the reference from which descriptions of these parameters are taken is the GPOPS-II® user's guide [27].

Table 3.2. GPOPS-II® Parameters used for HV Solution

Parameter	**Value**
Mesh Tolerance	$1x10^{-4}$
NLP Tolerance	$1x10^{-6}$
Derivative Stepsize	$1x10^{-8}$
Mesh method	hp-LiuRao-Legendre
Minimum number of collocation points	4
Maximum number of collocation points	30-50
Mesh sigma	0.5
SNOPT maximum number of iterations	1000
Derivative Supplier	Sparse finite differencing
Derivative Dependencies	Full
Scaling	None
Setup Method	RPM-Differentiation

The NLP tolerance provided Sparse Nonlinear Optimizer (SNOPT) with the allowable constraint violation tolerance. The mesh tolerance provided the desired accuracy for the discrete approximation of the optimal control solution. Once solved by the nonlinear programming (NLP) solver for a given mesh, the problem was untranscribed to a discrete approximation. An error for the discrete approximation was estimated and compared to the mesh tolerance and further mesh refinement was conducted if the tolerance was not met.

The *hp-LiuRao-Legendre* mesh refinement method was utilized since the method was observed to provide the most reliable convergence rate for the given problem formulation. *hp* mesh refinement methods use Gaussian quadrature orthogonal collocation at Legendre-Gauss-Radau points, and adjust both the mesh size and approx-

imating polynomial degree during refinement. it methods also reduce the degree of the approximating polynomial where the solution is smooth and reduce mesh density where the error tolerance has been met. The *hp-LiuRao-Legendre* method differs from other *hp* methods in the procedure used to determine whether an increase in the number of mesh points or an increase of the approximating polynomial degree is appropriate for further mesh refinement.

The *mesh sigma* parameter, in combination with the decay rate of the Legendre polynomial coefficient expansion, determines whether to increase the approximating polynomial degree within the interval or to create new intervals.

The mesh tolerance, the SNOPT tolerance and derivative step size were typically relaxed for troubleshooting and initial attempts at convergence. Values for these parameters during the solution process are discussed in Section 3.7.4.

The minimum number of collocation points was set to four, and the maximum number of collocation points between 30 and 50. The *SNOPT* NLP solver was utilized due to the sparse nature of the derivative matrices. The derivative supplier was typically set to *sparseFD*, specifying the forward differencing method, and derivative dependencies were set to *full*. No automatic scaling was used since the dynamics equations were previously scaled. *RPM-Differentiation* was used as the setup method.

3.7.3 Usage of the Hamiltonian

As discussed in 2.4 the Hamiltonian for the helicopter dynamic model was linear with respect to the controls. This implied that optimal solutions involved one or more singular arcs. Many solutions examined in this study resulted in Hamiltonians which were comprised of several semi-constant, discontinuous segments, as seen in Appendix A. This behavior was similar to results obtained by Rao for a problem involving singular arcs [1]. The minimization of the Hamiltonian value aided in selection of

tolerances, as described below.

3.7.4 Tolerance Selection and Convergence

During multiple solution attempts, it was noted that changes in the mesh tolerance and NLP solver tolerance affected the solution. A convergence study was conducted to determine the proper range of values for these two parameters. Optimal descents from the high hover point were examined with a weighted combination of initial altitude and final time for the objective function. The rotor tip path plane angle was bounded to $\pm 4.8\,\text{deg}$. The derivative step size was held constant at $1\text{x}10^{-8}$, all parameters from Section 3.7.2 were held constant and the initial solution guess was held constant as given in Tables C.1 through C.3. The initial altitude solution and the Hamiltonian were used to conduct the convergence study. The root mean square error of the Hamiltonian was used as an indicator of optimality. Semi-constant discontinuous segments and low values of root mean square error were assessed as indicators of appropriate tolerances.

The study showed that the largest satisfactory tolerance was $1\text{x}10^{-4}$ for both the mesh and NLP tolerances and that solutions became insensitive to mesh tolerance at NLP tolerances less than $1\text{x}10^{-4}$. Minimum values for root mean square error in the Hamiltonian coincided with convergence in the initial altitude solution. As each tolerance was decreased further, in various combinations, the optimal initial altitude changed very little (± 2 feet).

This study was only conducted at the high hover point, and was not meant to find a single tolerance combination for all runs. Rather, it provided a general idea of what tolerance values should provide reliable solutions. Generally, tighter NLP tolerances aided in reducing scatter in the Hamiltonian. Reduction of the mesh tolerance below $1\text{x}10^{-4}$ was mostly used to prevent GPOPS-II® from ending a run

without satisfying the constraints. For many solution attempts, both tolerances were set as tight as convergence would allow. For the majority of runs, the mesh tolerance, NLP tolerance and derivative step size were set to the values shown in Table 3.2. Full results from the convergence study are contained in Tables C.4 and C.5 in Appendix C.

3.7.5 Initial Guess Formulation

GPOPS-II® requires an initial guess of the solution in the form of at least two values for the time, states and controls. A very coarse initial guess was provided to GPOPS-II, composed of three values for each state and control. The solution was found to be quite sensitive to initial guess values, and heuristic techniques yielded initial guess matrices for the upper portion of the curve and the lower portion respectively. For the upper portion of the curve, an initial guess was required for each of the two phases. These matrices are depicted in Tables C.1 through C.3.

As compared to indirect methods, direct collocation typically requires much less knowledge of the solution. However, it was found that the GPOPS-II® solution to the HV problem was fairly sensitive to values in the initial guess, making changes in the initial guess a very coarse assumption of the solution form. The initial guess was altered for some data points to ensure GPOPS-II® converged on a realistic solution.

The usual technique of feeding solutions back into GPOPS-II® as a more accurate initial guess did not appear to yield better results. This technique caused additional oscillations in the controls and more scatter in the Hamiltonian.

3.7.6 Solution Technique

After values for the parameters listed in section 3.7.2 had been assigned, the entire HV was obtained by manually varying the initial airspeed condition. Iteration was

required on the starting altitude guess, since the initial guess value influenced the steady level flight trim initial condition if the initial altitude placed the helicopter model in ground effect. It was assumed that a solution that matched actual pilot technique was more desirable than a truly optimal solution, and that the overall shape of the single-engine HV diagram of the AH-1Z was similar to previously computed diagrams. Table 3.3 shows the basic solution sequence.

Table 3.3. Solution Process

	Step
1)	Set the initial airspeed
2)	Enter a guess for the initial altitude
3)	Set bounds for the initial altitude
4)	Set bounds for C_T, $\dot{C}_T$, α and $\dot{\alpha}$
5)	Adjust guess values for maximum vertical and horizontal velocity
6)	Adjust tolerances for the mesh, NLP solver and derivative step size if required
7)	Iterate until the guess for initial altitude matched the solution

An SNOPT exit flag of 0,1 (*successful finish, optimality conditions satisfied*), was verified for all solutions reported in this research. Solutions were further evaluated by checking the solution final time for a reasonable order of magnitude and comparing the final time against flight test data where possible.

3.8 Control Definition and Objective Function Study

A study was conducted by varying the control definitions and objective function. Control definitions were varied by using both the nine and eleven state models. The 11 state model allowed the accelerations of C_T and α to be minimized in the objective function. The objective function was varied with different combinations of initial altitude, final time and the accelerations from Eq. 3.33a and 3.33b. Solutions were evaluated based on control solution oscillation and scatter in the Hamiltonian. Solutions were examined at the high hover, knee and low hover points.

IV. Results

4.1 Chapter Overview

This chapter presents a validation of the dynamic model, comparison of an optimal control solution with flight test data and the method used to calibrate solutions with flight test data. The results of a study in control definition and objective functions are then presented along with the full Height-Velocity (HV) diagram for the AH-1Z. The chapter concludes with a comparison of induced velocity models.

4.2 Helicopter Model Validation

The helicopter model described in Chapter 3 was validated by using aircraft input parameters for the AH-1Z and comparing analytical results with flight test data obtained from [29] and [31]. Two methods were used. First, the power required for level flight was calculated using the dynamic model and compared against level flight performance data from [31]. Second, an open-loop simulation was used to compare analytically derived trajectories with flight test time histories.

4.2.1 Level Flight Performance

Level flight performance data were available for the AH-1Z in [31]. The helicopter dynamic model was used to calculate the referred shaft horsepower required for level flight with flight conditions matched to that shown on pages B-2 and B-3 of [31]. The *fsolve* function was used to solve equations 3.35a,3.35b, and 3.35c for energy-balanced values of C_T, α, and engine power. The power required for level flight was then calculated for the main and tail rotor. Speed power polars are shown in Figures 4.1 through 4.3.

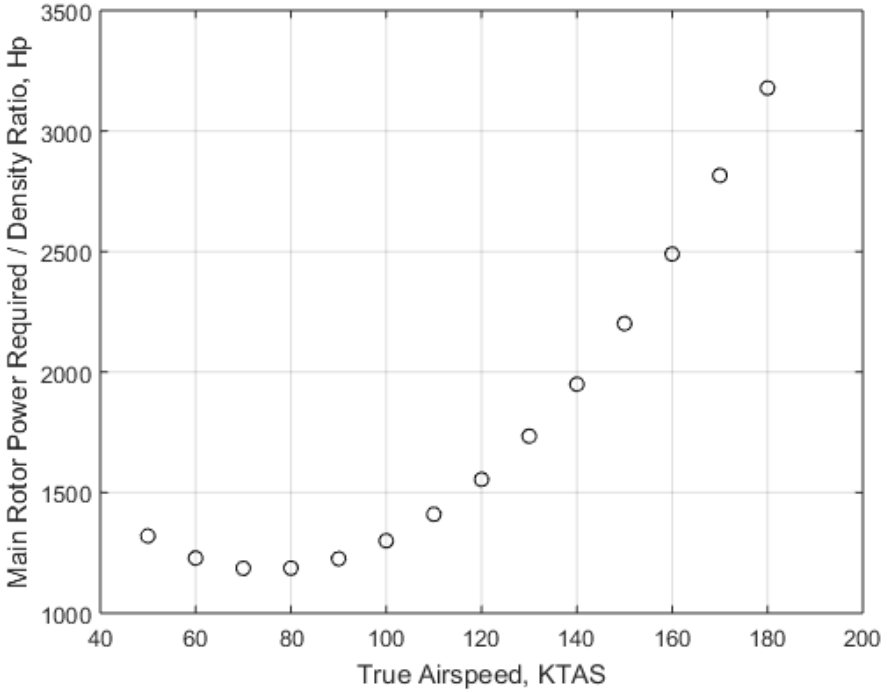


Figure 4.1. AH-1Z Main Rotor Referred Power Required

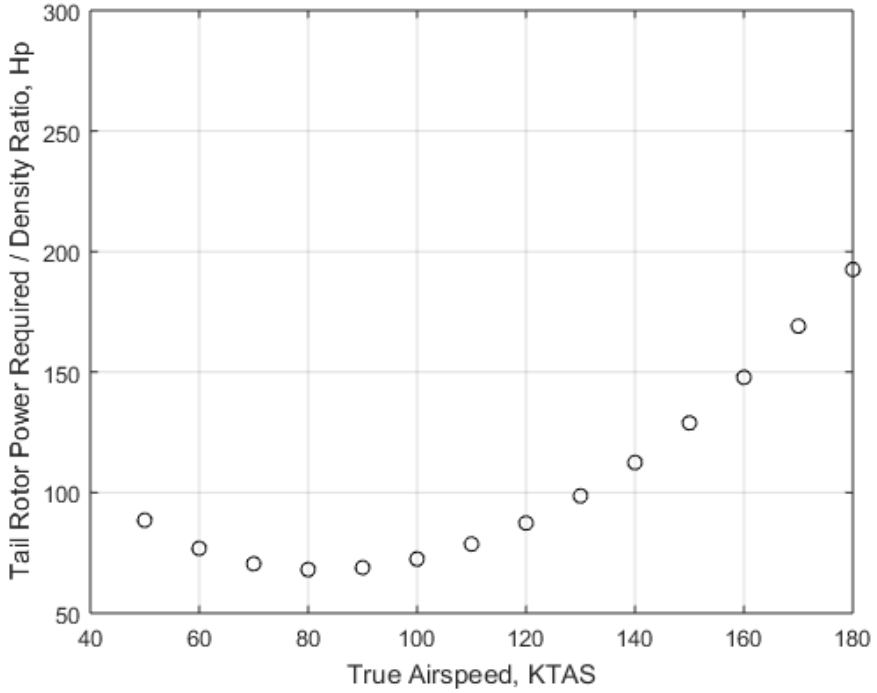


Figure 4.2. AH-1Z Tail Rotor Referred Power Required

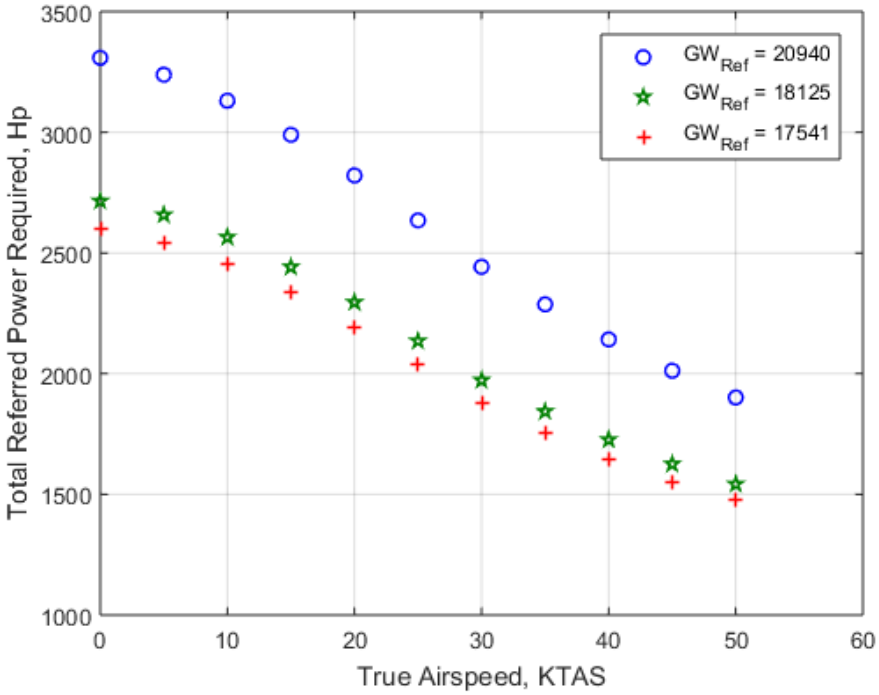


Figure 4.3. AH-1Z Referred Low Speed Performance

The dynamic model was calibrated to flight test data by adjusting the induced velocity correction factor, a flat plate drag correction factor added to the horizontal flat plate drag in the parasite power term, main and tail rotor efficiency factors and the μ term in the tail rotor power required equation. The best fit with flight test data was found with an induced velocity correction factor of 1.08, flat plate drag correction of 10 square feet, main rotor efficiency of 0.88, tail rotor efficiency of 0.7, and the $(1 - 1.2\mu)$ term shown in equation 3.37c. If Fig. 4.3 is compared with Fig. 7-67 in [29], good correlation is observed, with the helicopter dynamic model slightly overestimating the hover power required as compared to flight test data.

If Fig. 4.1 and Fig. 4.2 are compared with pages B-2 and B-3 of [31], it is observed that the dynamic model underestimates main rotor power required in the 50 to 70 knot range, and provides good correlation at higher airspeeds. The analytically calculated tail rotor power required diverges from flight test data at approximately 105 knots.

Since the contribution of the tail rotor to the total power requirement is small ($< 10\%$) and since the HV solutions under consideration did not involve airspeeds higher than 100 knots, the error was considered acceptable for this research.

4.2.2 Simulated Single-Engine Failure

Fig. 7-70 and Fig. E-73 through E-75 in [29] present a time history of a simulated single-engine failure from a low altitude, low airspeed condition. Included in Fig. 7-70 are values for the rotor thrust coefficient and rotor tip path plane angle which were derived by Carlson using the method shown in [18].

Data were manually extracted from the thrust coefficient and rotor tip path plane angle plots. These data were interpolated using a piecewise cubic method and were used as inputs (x_8 and x_9) in an open loop simulation. The dynamics equations were integrated using the single-step, explicit Runge-Kutta method algorithm *ode45*. The results of this simulation provided a means to assess the accuracy of the helicopter model used in this research by directly comparing dynamic, analytical results to flight test data depicted in [29]. Note that the convention for TPP angle is reversed from that used in [29]. Time histories for the simulated single-engine failure are shown in Fig. 4.4.

Overall, the dynamic response of the helicopter model to the inputs shown in Fig. 4.4 compares fairly with the flight test data presented on page 7-88 of [29]. Several discrepancies can be observed, but the most notable is in the sink rate. The sink rate is too high for portions of the trajectory. Also, a negative sink rate develops twice during the trajectory. It was assessed this was caused by direct modification of the C_T variable with the ground effect parameter.

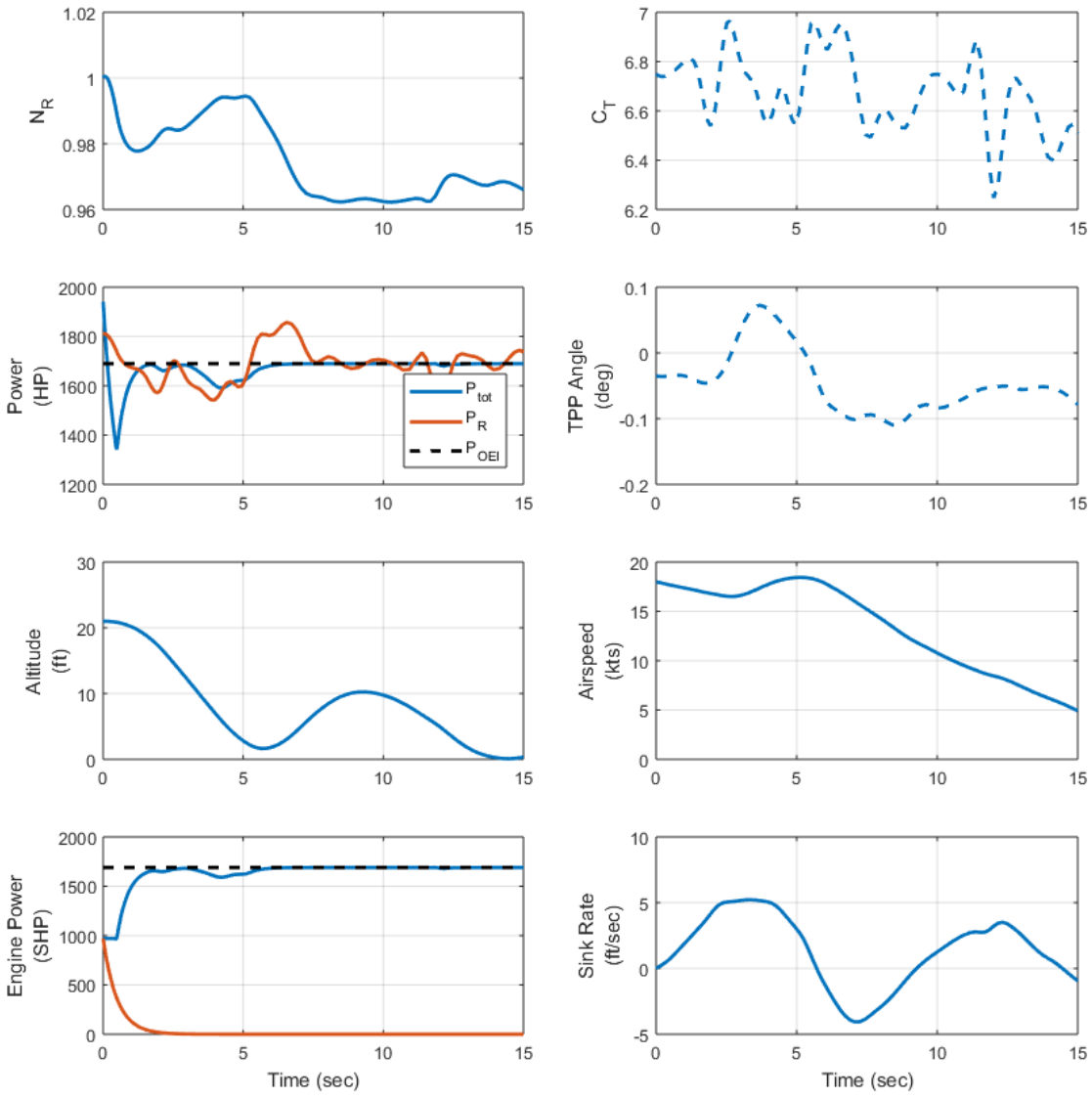


Figure 4.4. Open Loop Simulation of a Single-Engine Failure

4.2.3 Helicopter Dynamic Model Validation Summary

The dynamic model was insufficient for full flight simulation but was judged adequate for the purposes of this research due to favorable comparisons with level flight performance data and a simulated single-engine failure.

4.3 Comparison of Optimal Control Solution with Flight Test Data

By limiting the feasible set of initial altitude to the initial conditions for a flight test event, the optimal control solution was compared with flight test data. The same simulated single-engine failure flight test event used to validate the dynamic model in paragraph 4.2.2 was used for comparison with an optimal control solution. The 11 state model was used, the initial airspeed was set appropriately, and bounds were adjusted to match Fig. E-73 through E-75 of [29]. Figures 4.5 and 4.6 show time histories for the optimal controls and states respectively.

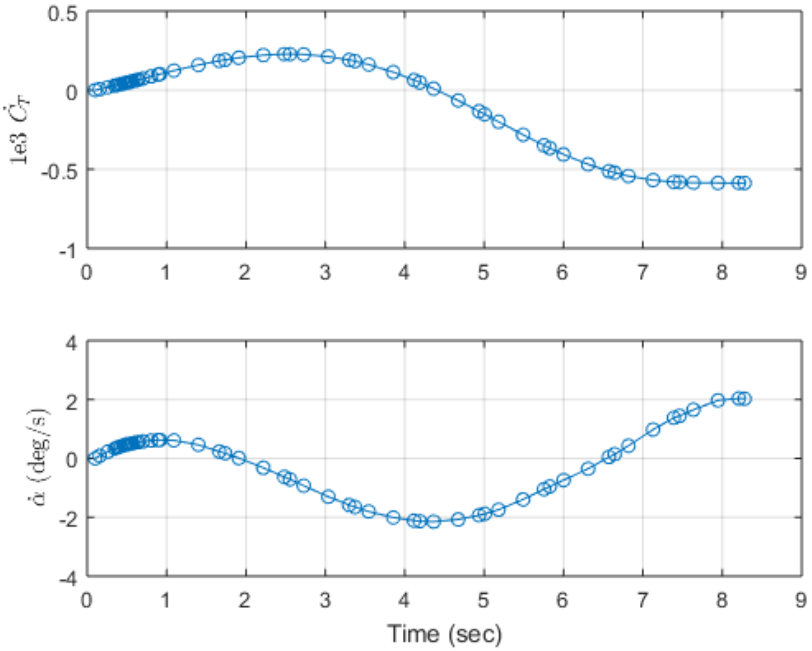


Figure 4.5. Optimal Controls for Single-Engine Failure at 20 ft, 18 knots

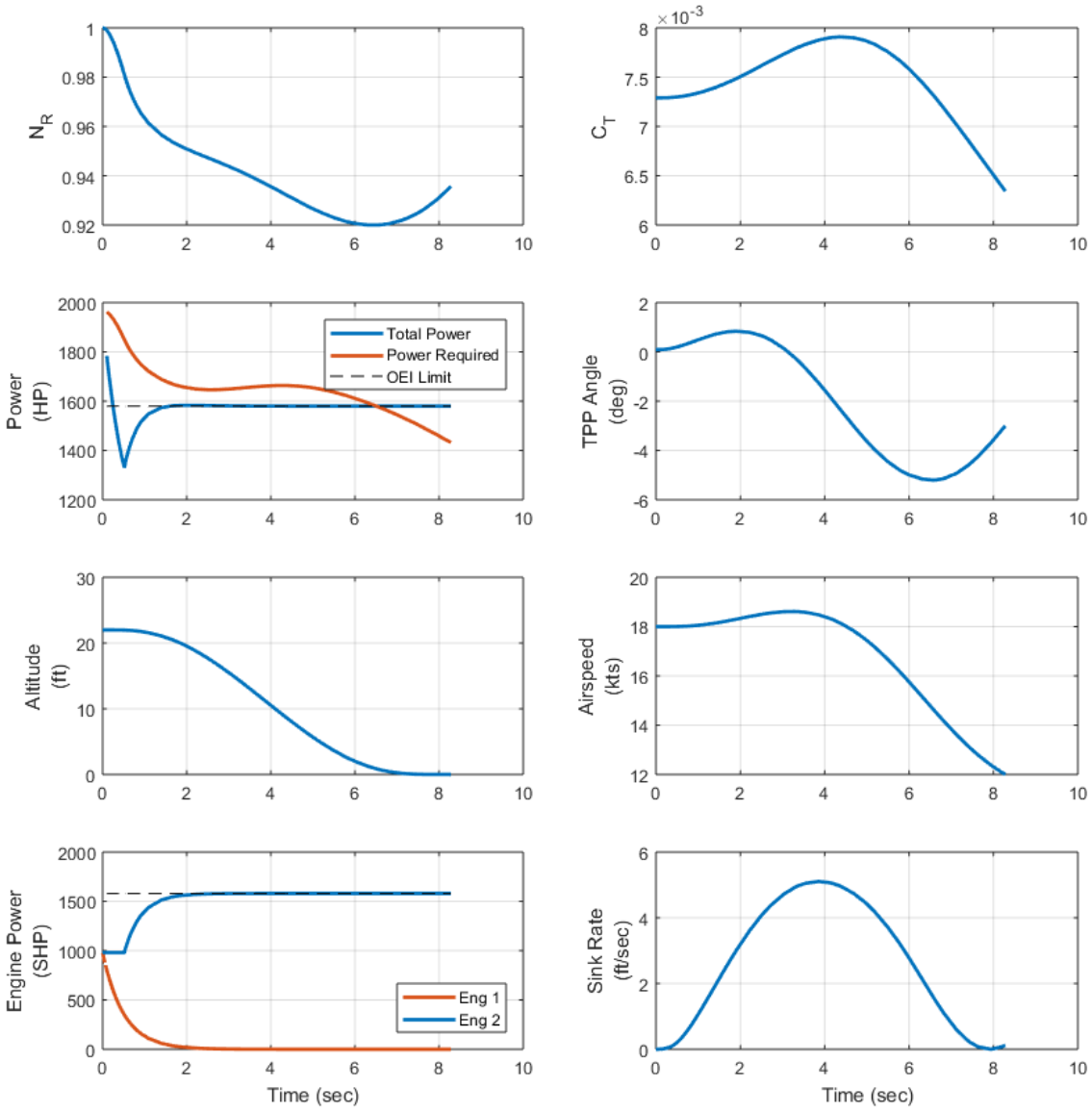


Figure 4.6. Optimal States for Single-Engine Failure at 20 ft, 18 knots

The optimal control solution has a different shape since the accelerations of C_T and α were being minimized. Overall though the airspeed, rate of descent, tip path plane and altitude time histories compare well. The final time of the optimal control solution 8.3 seconds is significantly shorter than the flight test touchdown time of approximately 15 seconds. This disparity in final time is likely due to a higher thrust coefficient used for a longer duration in the optimal solution.

4.3.1 Rotor Speed

A comparison of the rotor speed time histories highlights a key point about the problem formulation used in this research. The optimal control solution clearly droops the rotor to its lowest available value of 92%. In the flight test run, the rotor droops approximately 3.5% and then returns to its nominal value. The test pilot reduced the power demand (blade pitch) to maintain rotor speed during the simulated single-engine failure. The optimal control solution utilized a higher C_T causing the rotor speed to droop. This is caused by the dynamic optimization problem formulation. Since there is no pilot model included in the overall dynamic model, no mechanism exists to maintain the rotor speed at the nominal value when the power required exceeds the optimal solution for power demanded. The squared difference between rotor speed and nominal rotor speed was experimentally placed in the objective function but the results were inconsistent.

4.3.2 Hamiltonian

The Hamiltonian for the optimal control solution is shown in 4.7. Note the 10^{-5} scale. The near constant value of the Hamiltonian occurred repeatedly during this research when the bounds for initial altitude were set such that the h_0 term in the objective function could be minimized to zero. Essentially if the initial altitude bounds

were set so that GPOPS-II® solved for a local extremal vice a global extremal, then the computed Hamiltonian was nearly constant at zero. This behavior was useful when identifying the knee point.

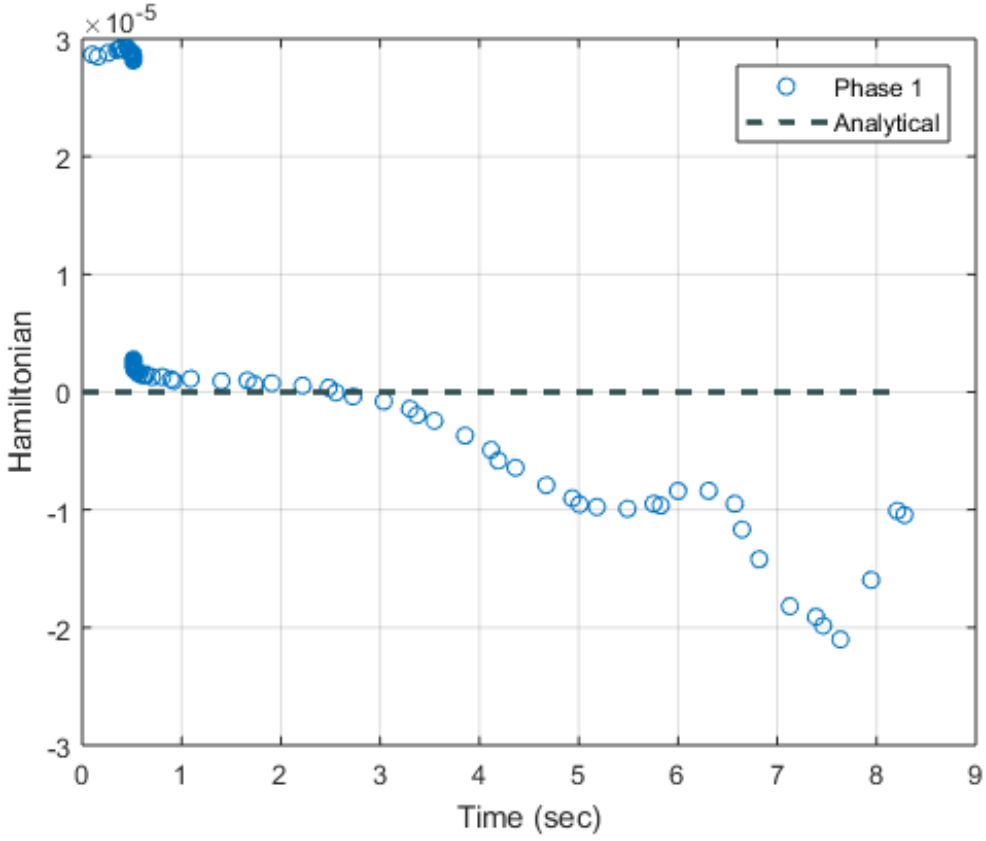


Figure 4.7. Optimal Hamiltonian for Single-Engine Failure at 20 ft, 18 knots

4.4 Calibrating the Optimal Control Solution Using Flight Test Data

Flight test data from [29] were used extensively to calibrate the model. When comparing flight test data with analytical solutions, test data for pitch attitude and pitch rate were considered approximations for rotor tip path plane angle and rate. For aircraft with different rotor system characteristics, a time lag may be necessary to keep this approximation within reasonable error.

As mentioned in Chapter 3, the initial altitude was quite sensitive to the choice

of bounds for several parameters. Most notably, the constraints on rotor tip path plane angle drastically affected the airspeed gained in a descent and therefore the aerodynamic efficiency of the descent (in terms of induced velocity). As an example of the sensitivity, if all other parameters were held constant and the bounds on tip path plane angle were altered from ± 15 deg to ± 5 deg the minimum initial altitude for a high hover changed from 98 feet to 501 ft. The natural course of action was to examine simulated single-engine failures, note the maximum pitch attitude and pitch rate, and use those values as bounds for the optimal control solution. In some cases however, this resulted in markedly different initial altitudes than observed in the flight test data.

A more dependable method for selecting appropriate bounds was to correlate the area under the optimal tip path angle solution with the area under the test data pitch attitude history. All solutions obtained with GPOPS-II® showed the same technique: decrease the tip path plane angle to the lower bound, hold it there as long as possible and then increase the tip path plane angle to decelerate the aircraft once in ground effect. The same control scheme was used by the test pilot: [29] shows that nose-low pitch attitudes and pitch rates were often rather pronounced. The difference lay in how long the nose was held at the extreme position. The pilot usually relaxed the nose-down attitude quickly so that the rate of descent did not reach an unrecoverable value. To account for different shapes of the two curves, the bound on tip path plane angle was adjusted to bring the two projected *areas* closer to the same value. Not to scale, Fig. 4.8 depicts this relationship. Bounds on rates and acceleration were drawn more directly from the data.

Also, an examination of flight test data shows that the pilot flew the aircraft differently for different flight conditions. At the high hover point, the initial pitch-down maneuver was quite pronounced. At the knee point and the low hover point,

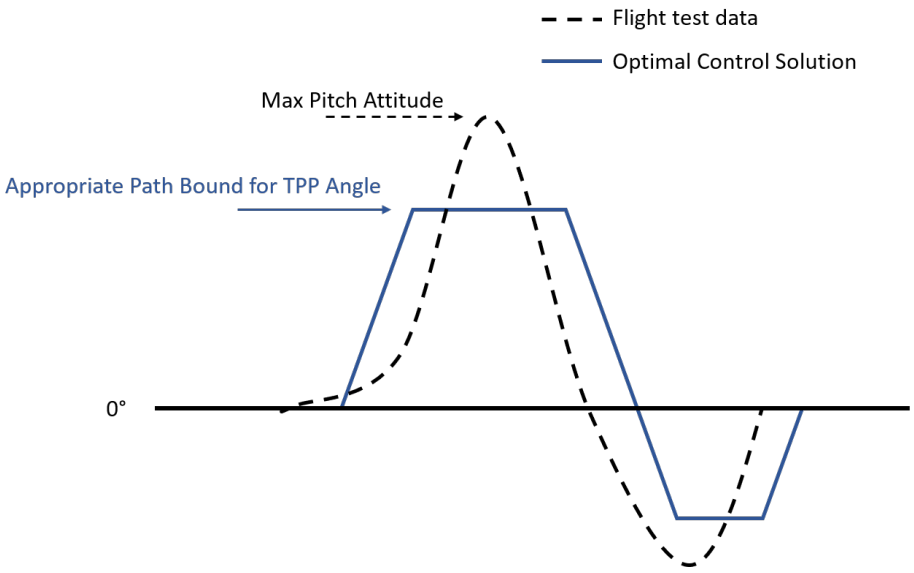


Figure 4.8. Choosing Appropriate Bounds for TPP Angle

the lowest pitch attitude was much less severe. This meant that bounds required adjustment to accurately determine the entire HV diagram. The attempt to match pitch angle time history areas was completed at the high hover point, low hover point and a location close to the knee point. After optimal control solutions were plotted at the three points, bounds were varied more or less linearly to obtain the remainder of the curve. Fig. 4.9 depicts the process. Bound selection was aided by the assumption that the curve should be smooth. For the 11 state model, bounds

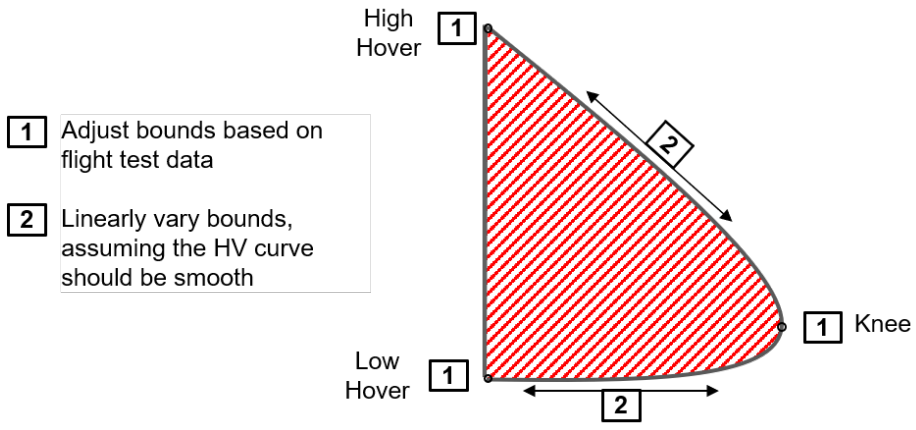


Figure 4.9. Calibration of Bounds

for $\ddot{C}_T$ were approximated using flight test pitch rate curves. This approximate value was used when possible, but at times had to be increased to achieve convergence.

The solution was far less sensitive to C_T and $\dot{C}_T$ bounds. The upper bound on C_T was never reached, and approximations for $\dot{C}_T$ from Carlson's Fig. 7-70 in [29] required no adjustment. These approximations were unchanged for each solution.

4.5 Control Definitions and Objective Function Study

One of the research objectives was to determine an avoid region which reflected not only the physical capabilities of the aircraft but also pilot technique. For example, the aircraft is capable of much steeper descents and much more aggressive flares than were observed in flight test time histories, so the HV avoid region could in theory be smaller. However, pilot workload is an important factor in determining the diagram. The HV diagram is not published to inform a pilot of the absolute boundary between a safe and unsafe descent which is only valid for a completely optimal control input sequence. Rather, the diagram informs the pilot of the boundary when using reasonable control inputs taught as standard pilot technique. A study was completed using optimal descent solutions from the nine state model and the 11 state model. Different objective functions were used to incorporate pilot technique and also add realism. Solutions are compared at the high hover point, the 10 knot point on the upper portion of the curve and at the low hover point. Time histories for the states, controls, Hamiltonians and mesh histories are available in Appendix A.

4.5.1 Nine State Model Minimizing Initial Altitude

Originally, a nine state model was used and initial altitude was either minimized (for the upper portion) or maximized (for the lower portion) using the following

objective function:

$$J = \pm h_0 \tag{4.1}$$

Two issues became evident in the solutions: first, the final time to conduct an optimal descent was excessive when compared to flight test time histories, and second, the control histories for the solutions demonstrate a phenomenon similar to rate limiting, with the controls instantaneously jumping from one bound to another. The time required for the optimal descent from 131 feet was approximately 35 seconds. Comparing with Fig. E-40 through E-42 from the AH-1Z HV test report [29], the flight test helicopter reached the ground in approximately 22 seconds. The oscillations in the control solution is shown below in Fig. 4.10. It was assessed that the

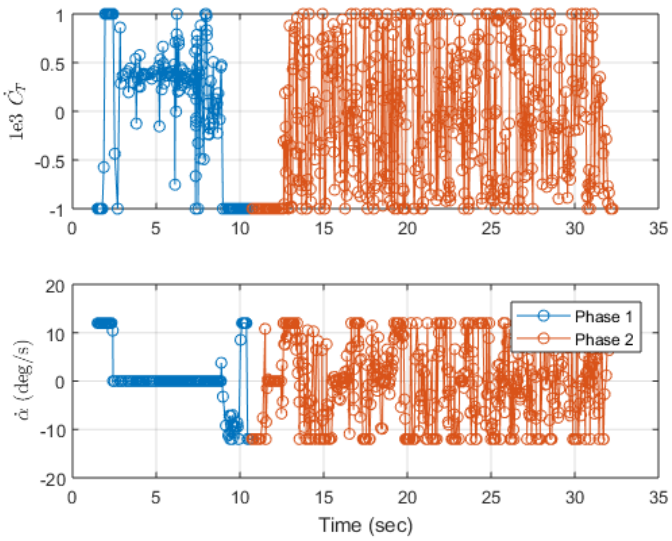


Figure 4.10. Optimal Control History Using 9 State Model

dynamic model engine governor moved out of phase with small changes in the power required, resulting in large C_T changes which exacerbated the oscillations and caused the rapidly fluctuating control histories shown in Fig. 4.10.

4.5.2 Nine State Model Minimizing Altitude and Weighted Final Time

The nine state model was improved by including weighted final time in the objective function, as shown in Eq. 4.2. Since the optimal control solution for this formulation did not purely shrink the avoid region, it stands to reason that initial altitudes using this objective function would be more conservative than those of the solutions described in 4.5.1. For the high hover point, the final time was 28 seconds, compared to the flight test time of 22 seconds. Strangely, the addition of final time in the cost function resulted in a less conservative initial altitude for the high hover, 10 knot point and low hover point. Control oscillations were still evident.

$$J = \pm h_0 + W_t \tau_f \tag{4.2}$$

When bounds on rotor tip path plane angle and tip path plane rate were adjusted appropriately, this formulation yielded initial altitudes that compared well with flight test data. However, to demonstrate feasibility, means were sought to reduce the oscillations in the $\dot{C}_T$ and $\dot{\alpha}$ solutions. Additionally, the single phase version of this solution method used to determine the lower portion of the HV curve did not converge above five knots.

4.5.3 Eleven State Model Minimizing Altitude and Weighted Controls

The 11 state model displayed superior convergence and produced much more realistic controls. Full solutions are shown in Appendix A. For a modest cost in computation time, oscillations in the control solution was substantially reduced, using the objective function shown in Eq. 4.3.

$$J = \pm h_0 + \int_{t_0}^{t_f} \left(W_1 u_1^2 + W_2 u_2^2 \right) dt \tag{4.3}$$

4.5.4 Comparison of the Solution Methods

A comparison of optimal control results using the same bounds in three different techniques is shown in Fig. 4.11. Significant differences are observed in the solutions for the low and high hover heights. At the high hover, the 11 state model altitude is nine feet above the nine state solution. At the low hover, the 11 state model altitude is 4.5 feet below the nine state altitude. The maximum difference in solutions at the 10 knot point is two feet. Although the nine state model with minimum final time provided the closest match to flight test data, by adjusting the bounds, either of the other two models could provide a closer match using different bounds. Fig. 4.11 demonstrates that although the control solutions are questionable, the nine state model still provides initial altitude solutions which are comparable to those obtained with more realistic control solutions.

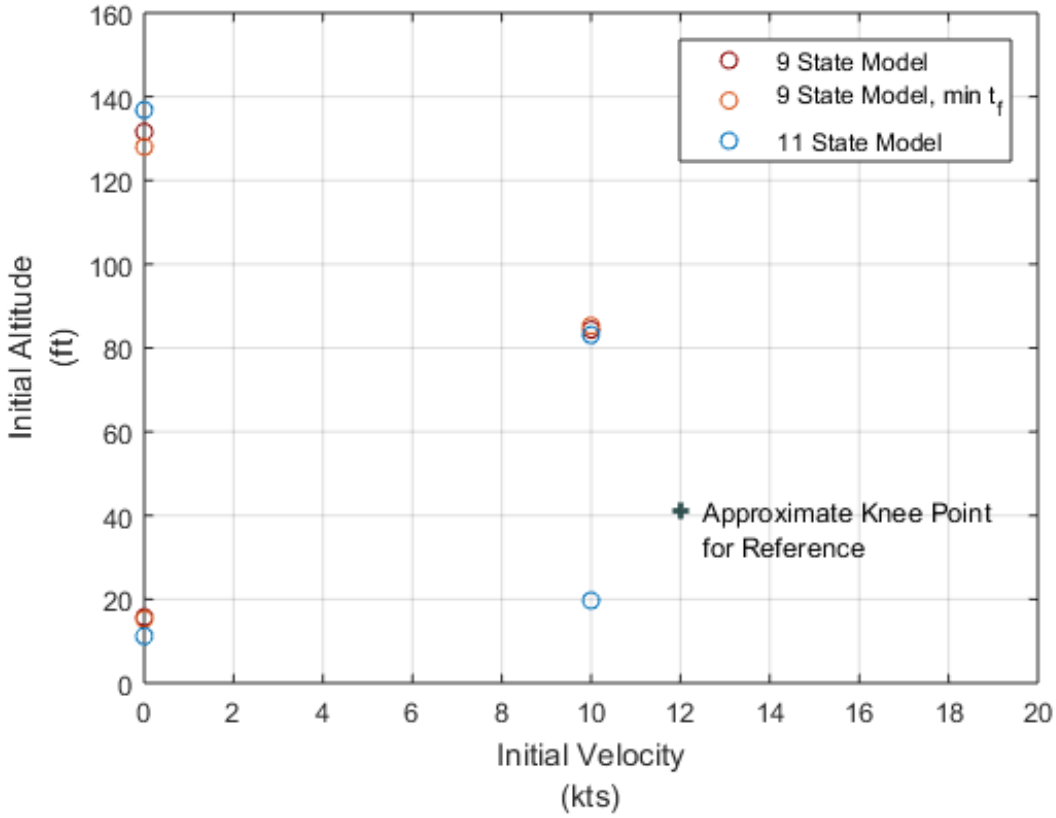


Figure 4.11. Comparison of Solutions from Different Optimal Control Formulations

4.6 Full HV Diagram for the AH-1Z

The HV diagram computed using the 11 state method is shown in Fig. 4.12. Included in the diagram are the points determined using the nine state, weighted final time formulation. The gap in the latter data set shows the area below the knee point where convergence was not attained.

4.6.1 Shape of the HV Diagram

Compared to the diagram obtained by Carlson in [29], the HV diagram in 4.12 defines a smaller avoid region. The high hover point in 4.12 is lower, the low hover point compares well to that of [29] and the knee point is less conservative. Also, the diagram in 4.12 changes slope more frequently, both above and below the knee. This

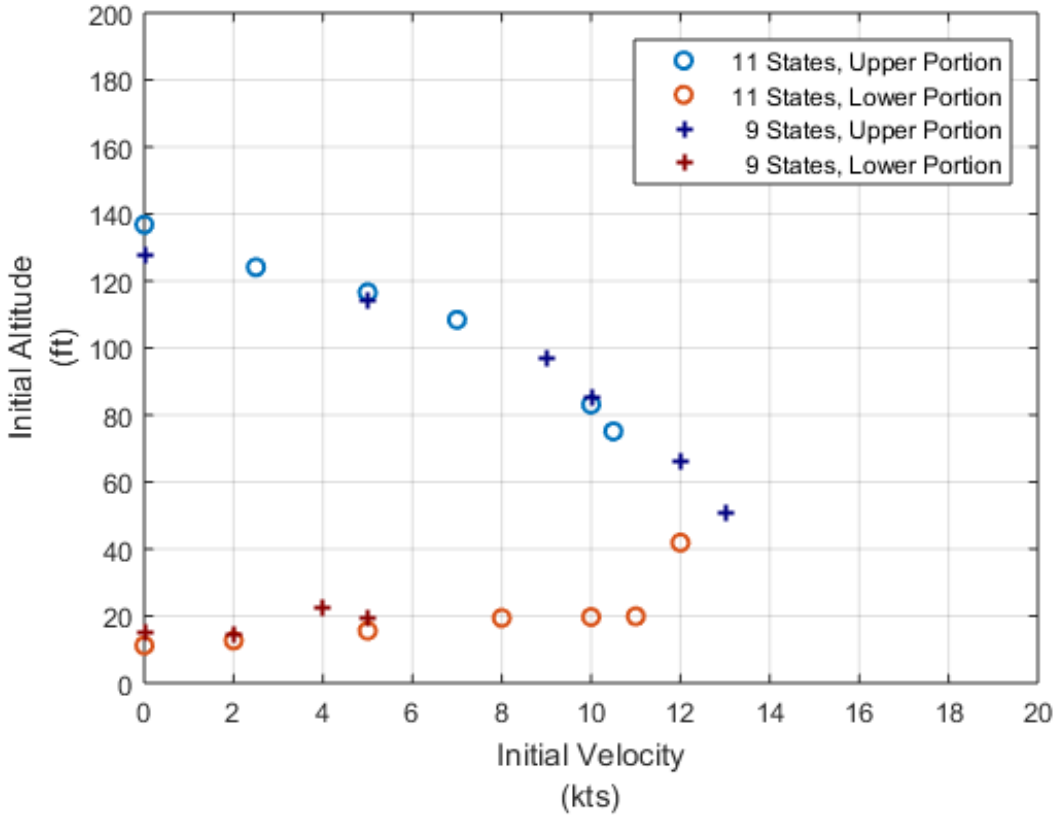


Figure 4.12. Resulting HV Diagram for the AH-1Z

is due to changes in the bounds on tip path plane angle and rate. It was noted during development of the diagram that if the bounds were held constant, the diagram was more linear in nature (though with very different values).

The increase in altitude for the lower portion of the curve in 4.12 may be caused by inaccuracies in the dynamic model. Specifically, the ground effect factor which directly modifies C_T was calculated with no dependence on advance ratio. Johnson [4] stated that ground effect is reduced as forward velocity is increased. Reduced ground effect for the lower portion of the HV curve may have the effect of flattening this portion, similarly to Carlson's diagram.

4.6.2 Identifying the Knee Point

Convergence was problematic in the vicinity of the knee point, for both the upper portion method and lower portion method. Bounds on the tip path plane angle and rate were continually reduced and convergence was attained at a 12 knot point using the single phase formulation for the lower portion.

The bounds used for rotor TPP plane angle and rate were reduced even further to $\pm 5\,\text{deg}$ and $\pm 5\,\text{deg}$ per second respectively. Additional solutions were attempted with the initial airspeed set to 13 knots (using the single phase method) and 14 knots (using the two phase method). Both methods were able to minimize the h_0 portion of the objective function to zero, with a corresponding near-zero, near-constant Hamiltonian. This indicated that no restriction existed at the initial airspeed. The bounds on initial height which were used are depicted in Fig. 4.13. At the airspeeds shown, this indicates that a safe landing is possible if the pilot exceeds the bounds listed above. This further shows that while the knee may not be well defined by numerous converged solutions, it does not extend past 14 knots.

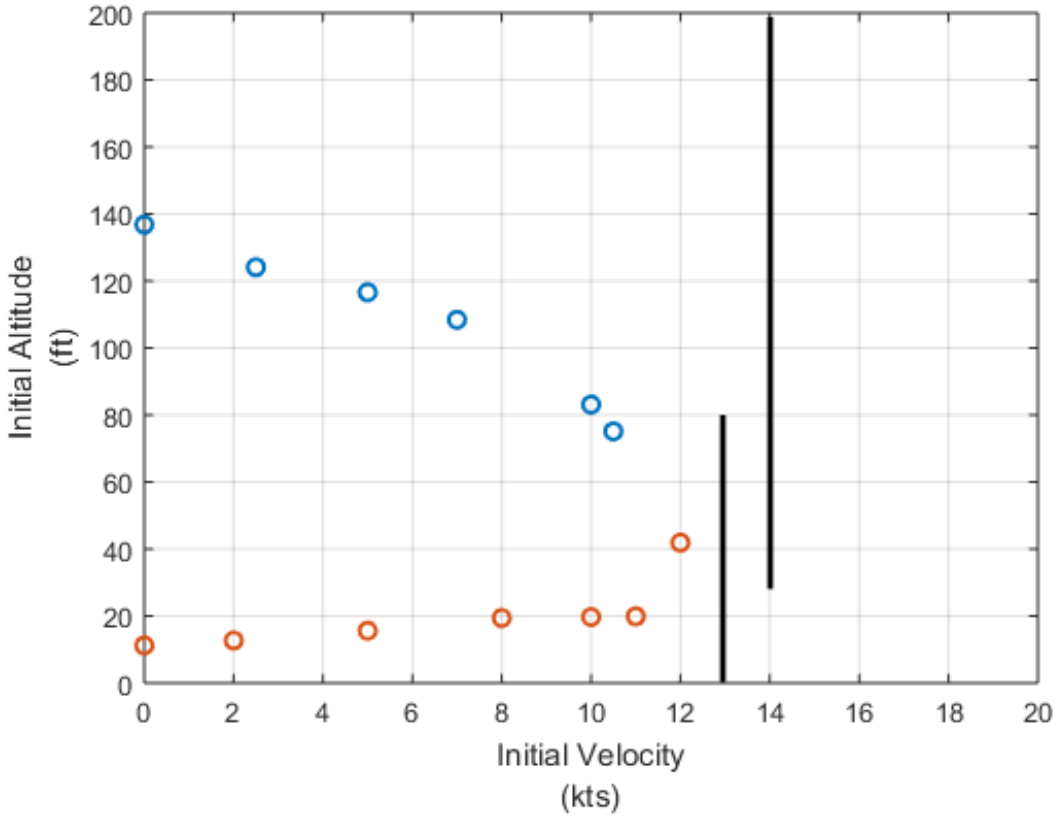


Figure 4.13. Determining Where the Knee is Not

4.7 Induced Velocity Investigation

The method of induced velocity calculation developed by Johnson in [2] was subsequently used in [24], [19], [21] and [29]. The results of the current research are presented using an updated calculation for induced velocity published by Johnson in [23]. Johnson developed the new model to account for the effects of vortex ring state (VRS). As can be seen in Fig. 4.14, the new model calculates a much wider range of induced velocity than the old model. Note that the curves labeled Baseline and VRS are all part of the new model. The dashed lines depict calculations using the old model. The continuous function used by GPOPS-II® was altered to calculate induced velocity using the old method for comparison with the results of this research which utilized the newer method. Comparing results for the high hover point, data

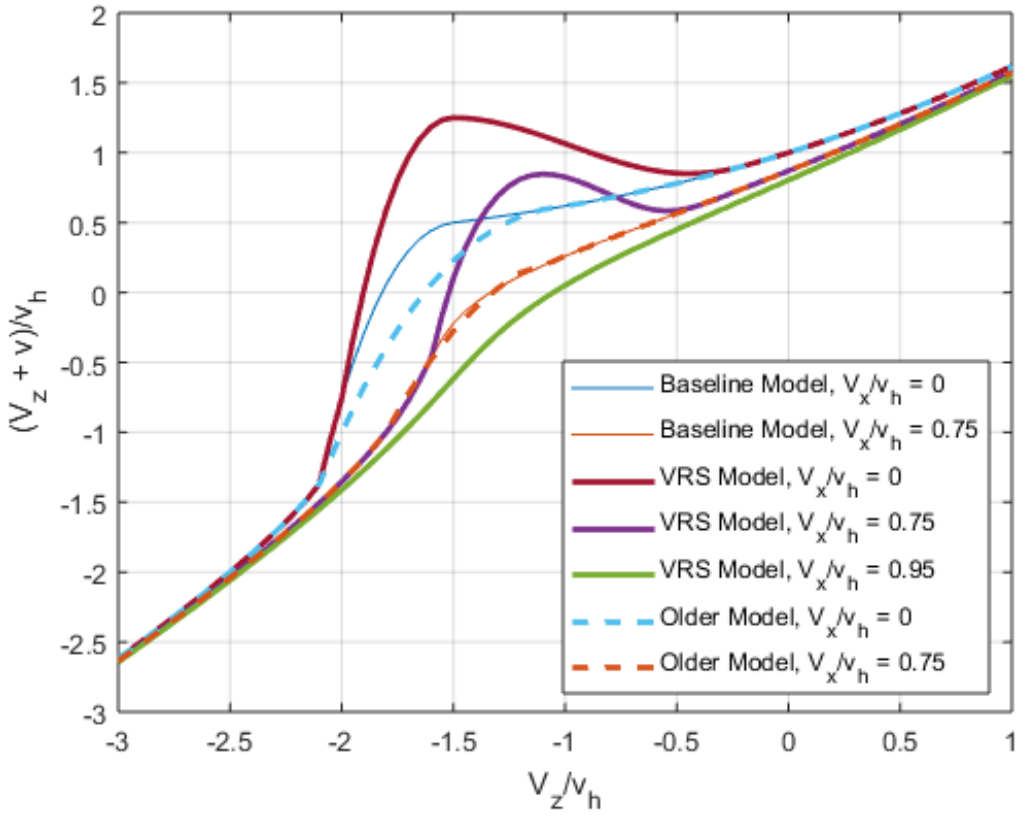


Figure 4.14. Comparison of Induced Velocity Models

are presented in Figures 4.15 and 4.16. Clearly the solutions for rotor inflow, and the full optimal control solutions differ from each other. (The difference in initial altitude was 5 feet). However, these plots do not tell the full story, since GPOPS-II® operated differently with the different models. The algorithm was free to avoid VRS when using the new model and therefore it cannot be determined whether VRS conditions were definitely present.

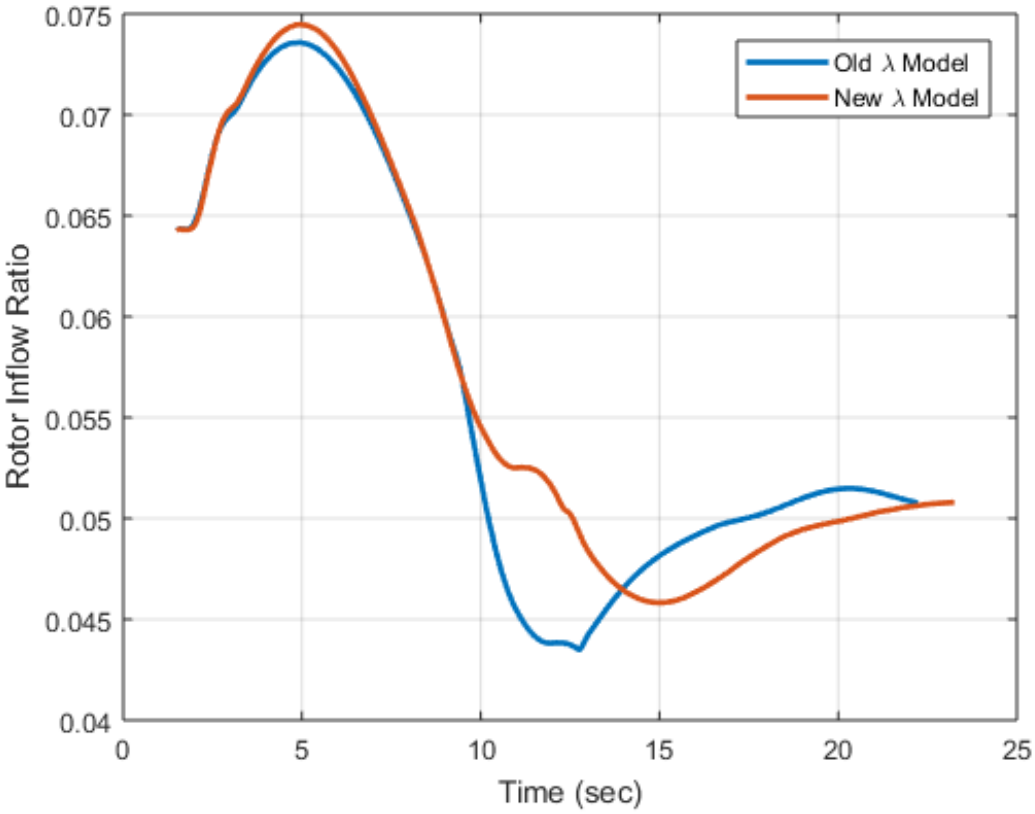


Figure 4.15. Optimal Control Rotor Inflow Comparison

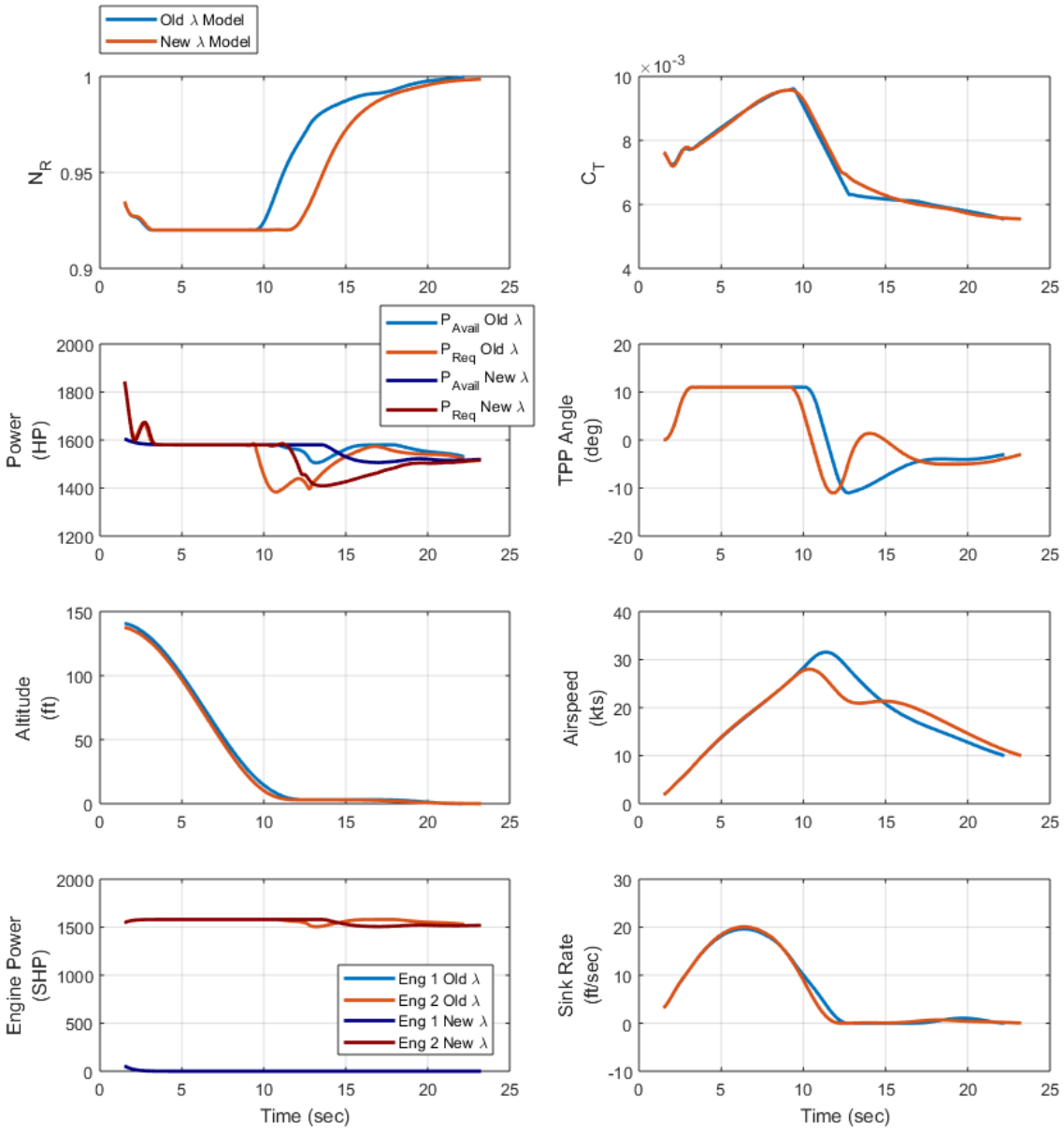


Figure 4.16. Optimal Control Solution Comparison

For a valid direct comparison, the optimal C_T and α calculated from the high hover using the old model were integrated with *ode45* using the new rotor inflow model in a closed loop simulation. Results are shown in Figures 4.17 and 4.18. As may be observed in Fig. 4.17, higher inflow is calculated using the new inflow model for the initial portion of the descent, but as the horizontal velocity increases, the two solutions vary considerably. The results of the closed-loop simulation vary considerably from the optimal control solution. It may be deduced from this comparison that the more accurate rotor inflow model of [23] affects the HV solution considerably.

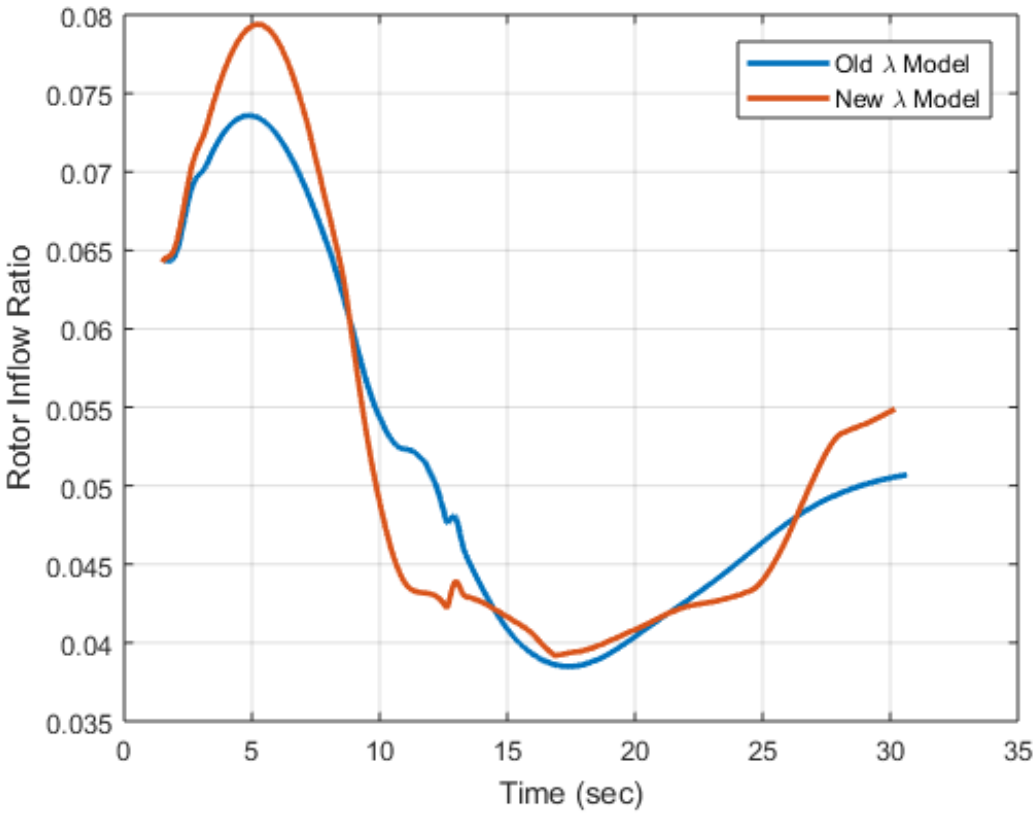


Figure 4.17. Optimal Inflow with Old Inflow Model Integrated using New Inflow Model

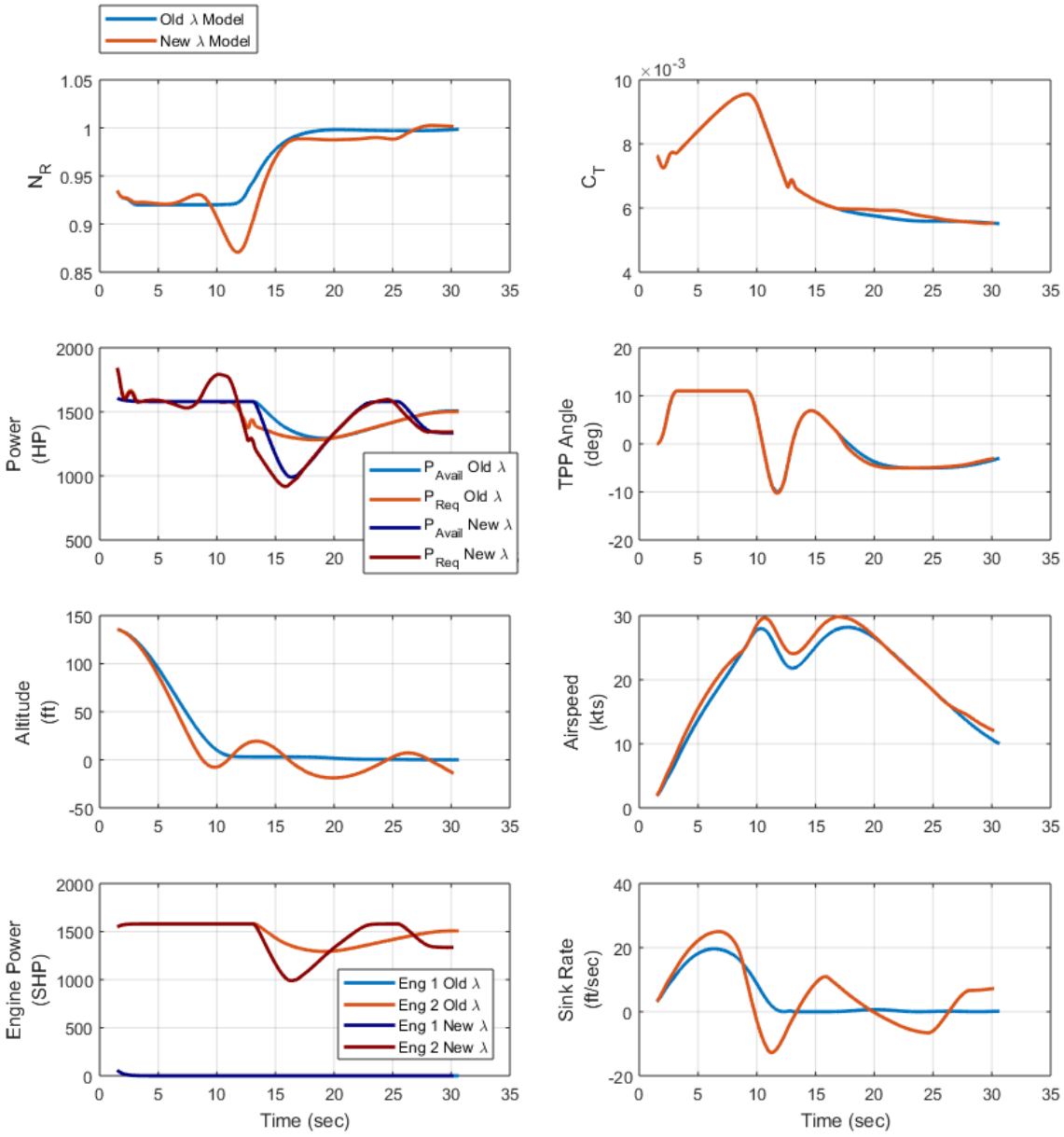


Figure 4.18. Optimal Solution with Old Inflow Model Integrated using New Inflow Model

V. Conclusions and Recommendations

5.1 Conclusions

The major objectives of this research, as stated in Chapter 1, were accomplished. A helicopter dynamic model was developed and validated. GPOPS-II® was used to solve an optimal control problem of the helicopter dynamic model in a single-engine failure scenario. Multiple iterations of this solution determined the entire Height-Velocity (HV) diagram. The effects of pilot technique were incorporated using state constraints. The solution procedure is easily adaptable to different platforms, by tailoring the parameters listed in Appendix E. Finally, the effect of different induced velocity calculation methods was examined. Conclusions from these research objectives are summarized below.

5.1.1 Helicopter Model Validation

The helicopter model described in Chapter 3 was validated using level flight performance data and flight test time histories. Efficiency factors for the main and tail rotor, an induced velocity correction factor and a flat plate drag correction factor were sufficient to adjust the model to match level flight performance data. Resulting open-loop simulations were satisfactorily compared with flight test time histories. The new method used to calculate rotor inflow had a noticeable effect on the trajectories. It is assessed that the rotor inflow method partially accounted for differences from Carlson's results [29].

5.1.2 Solution Using GPOPS-II®

GPOPS-II® was successfully used to solve for an optimal helicopter descent. This optimal descent included two phases, a free initial condition, open final time,

inequality constraints placed on all variables, unknown singular arcs, and a partially constrained final condition. With inputs matched to previous methods, the HV solution using GPOPS-II® was significantly less conservative. The previous method of Carlson [29], used SNOPT with a fixed collocation interval as compared to the adaptive mesh methods of GPOPS-II®. It is possible that the adaptive meshing of the GPOPS-II® software partially accounted for differences from Carlson's results.

The optimal control problem was based on minimizing initial altitude when computing points above the knee of the HV curve, or maximizing the initial altitude when computing points below the knee. This formulation prevented iterations on initial airspeed, but caused problems with convergence in the vicinity of the knee point. By tightening bounds on certain states, convergence was achieved near the knee, but precise definition of the knee point was not possible.

5.1.3 Nine State versus Eleven State Model

Results from a nine state model were compared with results from an eleven state model. The nine state model utilized the first derivatives of thrust coefficient and tip path plane angle as controls. The 11 state model used the second derivatives of the same variables as controls. The 11 state model allowed minimization of the controls, and produced much more feasible solutions for the thrust coefficient and tip path plane angle rates, as compared to the nine state model. However, the initial altitudes calculated by the two programs were similar. The 11 state model is recommended due to the improved control solutions.

5.1.4 Proper Adjustment of Bounds

Accurate solution methods did not automatically equate with solutions which matched flight test results. Proper selection of path constraints, in the form of bounds

on the controls and states, was crucial in achieving optimal initial altitudes that compared with those determined through flight test. Proper bounds represented not only physical characteristics and limitations of the aircraft but also pilot technique. A technique of matching areas under pitch attitude time history curves to the area underneath optimal control tip path plane angle trajectories was successfully used to match flight test data.

To utilize the methods developed by this research, detailed knowledge must be obtained for how the aircraft is flown during a single-engine failure. This is best obtained from flight test time histories, but detailed descriptions of simulated single-engine failures by qualified pilots may be sufficient.

5.1.5 Usage of the Hamiltonian

GPOPS-II® provides co-state estimates which were used to approximate the Hamiltonian. The Hamiltonian was useful during this research for selecting appropriate tolerances during a convergence study and for identification of the HV diagram knee point.

5.2 Recommendations for Future Improvements

The results obtained in this research could be improved by changes in the aircraft model, a better method for providing an initial guess, incorporating pilot technique in the cost function and by examining the solution method using different density altitudes and different aircraft. These recommendations are discussed below.

5.2.1 Improvements to the Dynamic Model

The ground effect calculation used in this research was only valid in hovering flight and did not include any dependence on forward velocity. This meant that ground

effect was overestimated at higher airspeeds. The airspeeds reached during solutions were not extreme, but incorporation of a μ term into the ground effect calculation would improve the results.

The power required for the main and tail rotor was calculated using momentum theory modified with certain elements of blade element theory. Blade element theory could be used entirely. Table lookups with rotor airfoil tables and numerical integration of the rotor disk would incorporate the effects of compressibility, blade stall, blade twist and changes in blade shape from root to tip. This would also enable the control to be changed from thrust coefficient to blade pitch angle, more accurately modeling the pilot-aircraft system.

5.2.2 Initial Guess

The GPOPS-II® program requires an initial guess for the solution of each state and control trajectory. A fairly coarse guess was used in this research, and was only scaled by altitude. A better process to provide guesses for the algorithm would likely result in more accurate, predictable solutions. Particle swarm optimization [35] and genetic algorithms [36] both hold promise to improve initial guess generation.

5.2.3 Automatic Bound Adjustment

As described in Section 5.1.4, the method of matching areas under pitch attitude time history curves to the area underneath optimal control tip path plane angle trajectories may be automated by incorporating the difference between the two areas into the cost function. With an appropriate weighting factor this addition could potentially yield the same results attained in this research without adjustment of constraints at each point on the HV diagram.

5.2.4 Effects of Referred Gross Weight and Different Aircraft

Using the methods of this research, an optimal control solution can be calibrated with flight test data. The method may be utilized to provide HV solutions which were not completed in flight test. Before this can be accurately accomplished, further investigations are required. The method described above for determining appropriate state and control bounds should be investigated at different referred gross weight conditions as this research considered only a single condition.

Finally, further validation of the methodology is required using one or preferably multiple different aircraft. It my be that implicit assumptions used on the case study aircraft do not apply to all platforms.

5.3 Summary

The current research constructed a dynamic model for a traditional helicopter which incorporated an improved induced velocity calculation. The dynamic model was used to frame a multi-phase, open final time, partially constrained initial state, partially constrained final state optimal control problem. The GPOPS-II® software suite was used to solve for the optimal initial altitude at given initial airspeeds. The result of this procedure was an analytically determined HV diagram for the AH-1Z Cobra at a single gross weight and single density altitude. This research provided the Naval Air Systems Command with a method to analytically calculate HV diagrams for traditional helicopters and provided modest advancements in the field of HV determination.

Appendix A. Selected Results for the AH-1Z

A.1 High Hover

A.1.1 Nine State Model, Minimizing Altitude Only

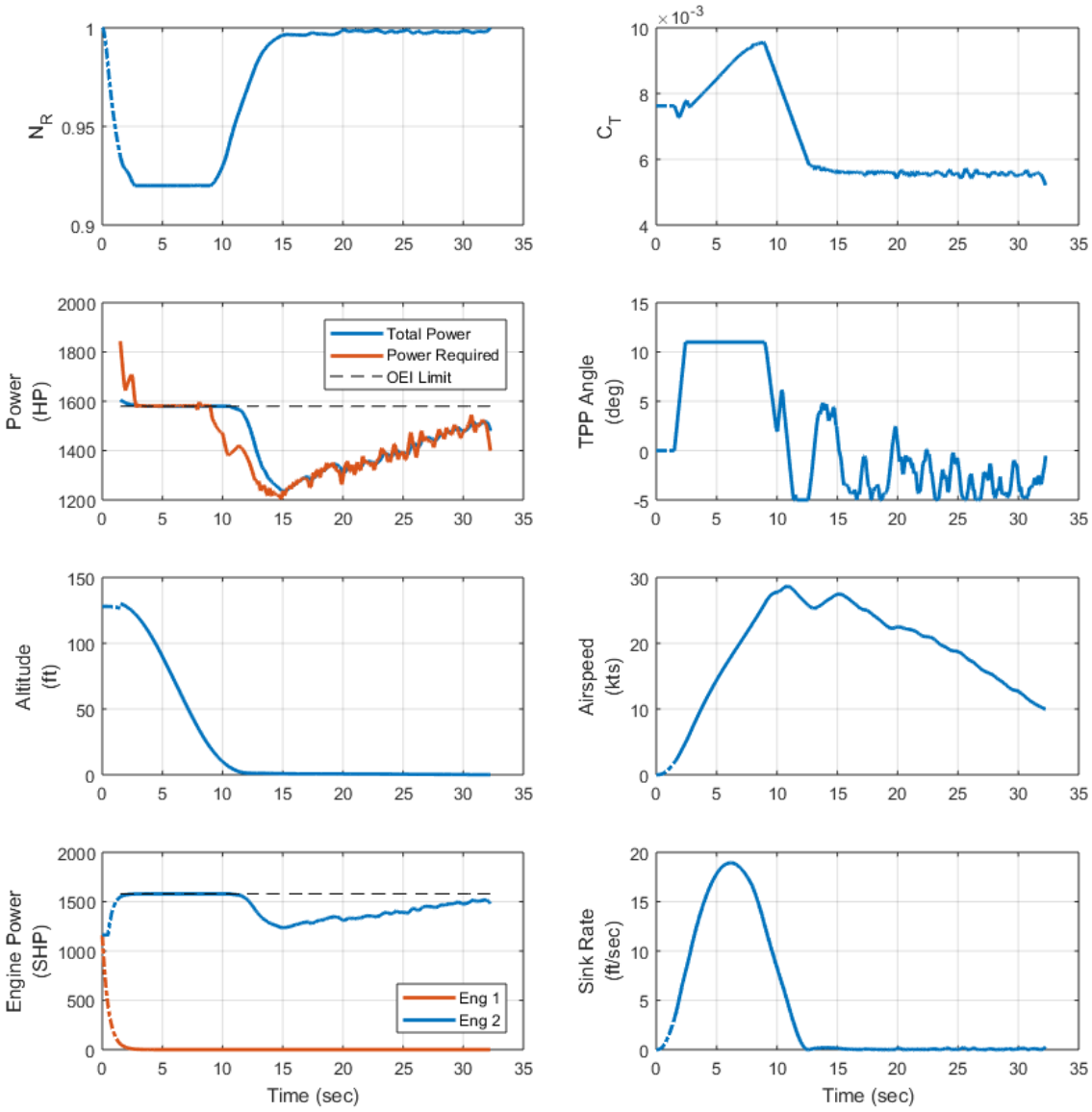
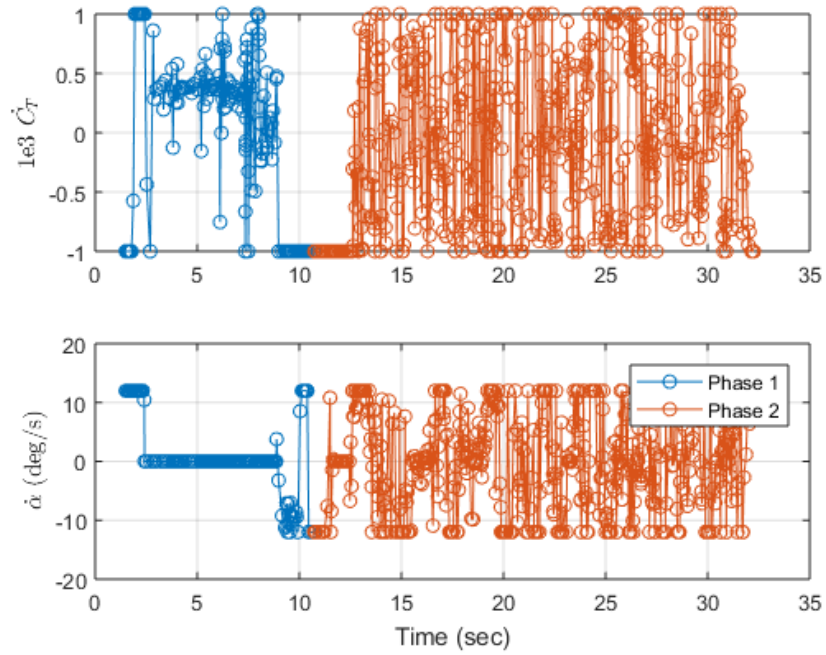
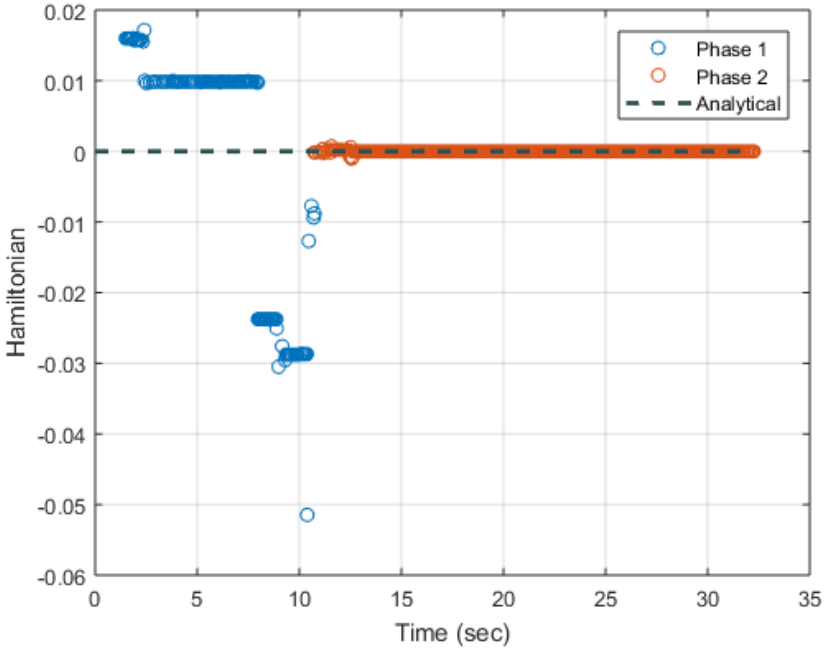


Figure A.1. High Hover, 9 State Model, $J = h_0$



(a) Controls



(b) Hamiltonian

Figure A.2. High Hover, 9 State Model, $J = h_0$, Controls and Hamiltonian

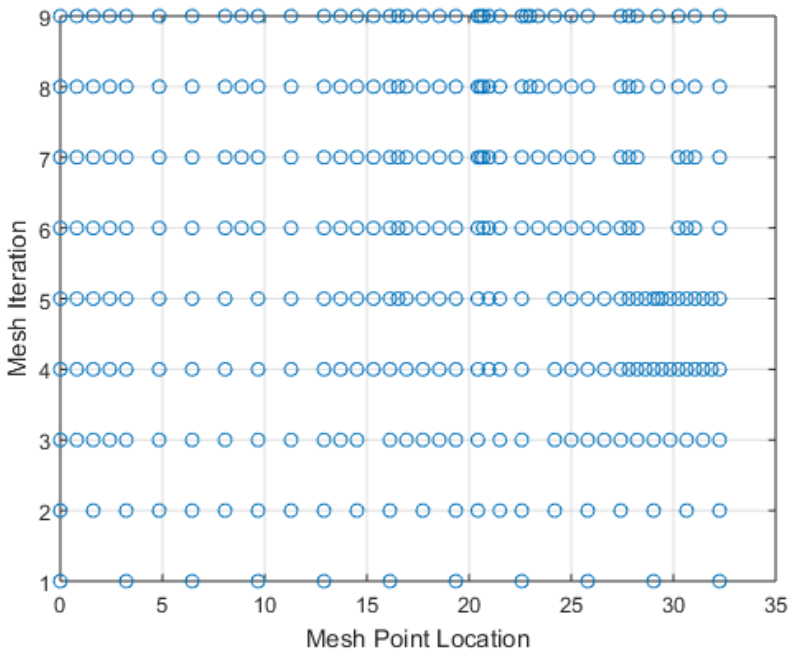


Figure A.3. High Hover, 9 State Model, $J = h_0$, Mesh History

A.1.2 Nine State Model, Minimizing Altitude and Final Time

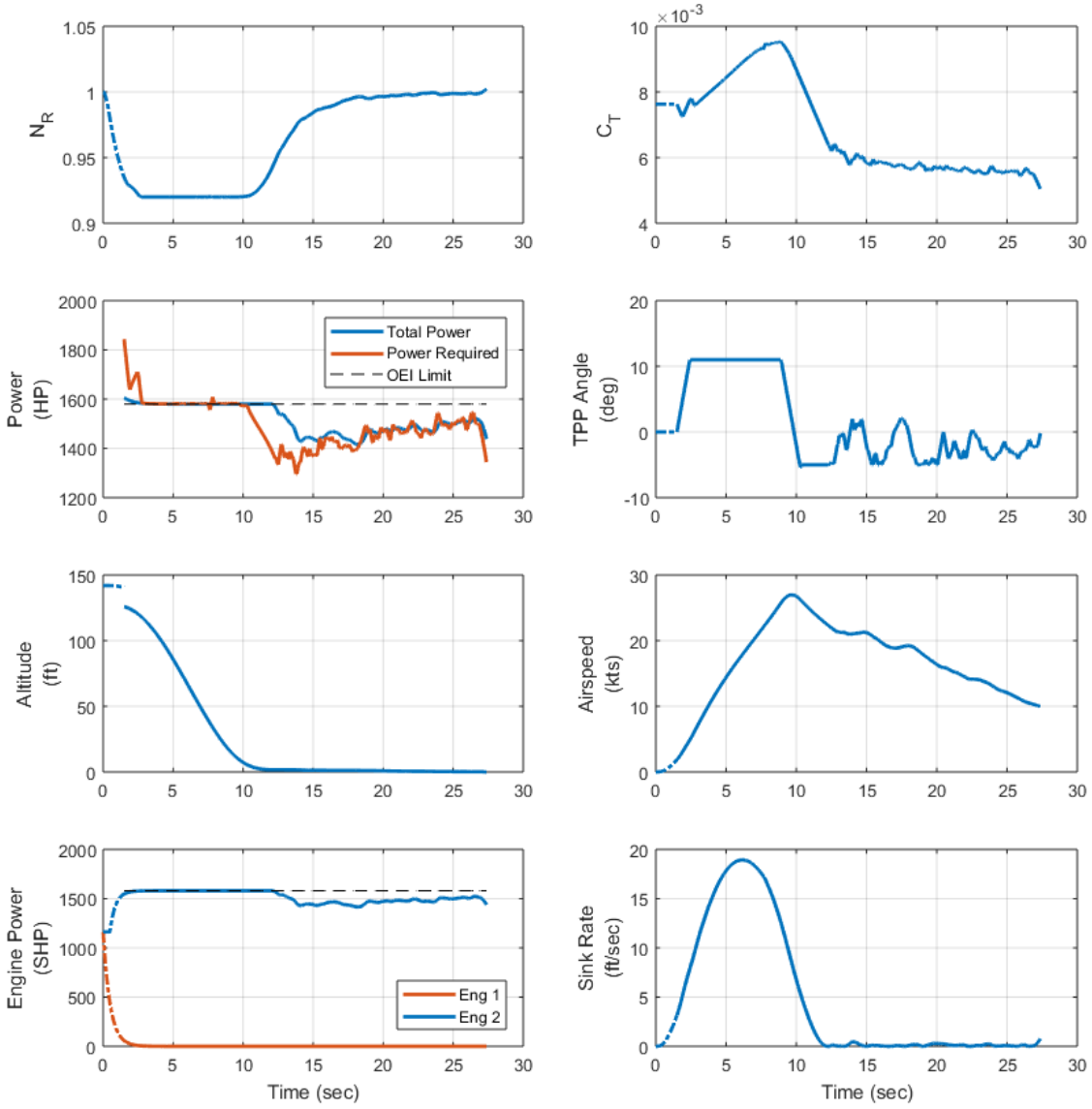
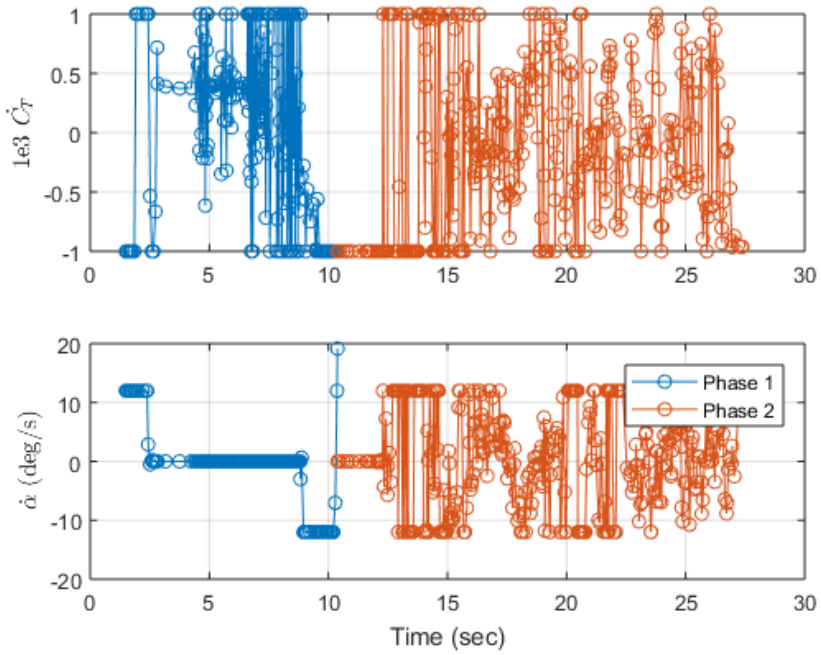
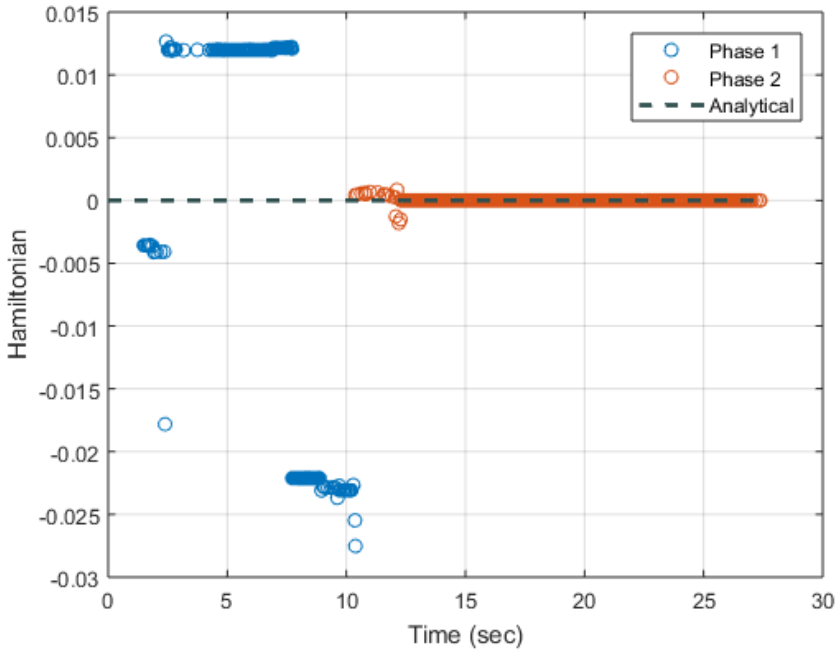


Figure A.4. High Hover, 9 State Model, $J = h_0 + W_t\tau_f$



(a) Controls



(b) Hamiltonian

Figure A.5. High Hover, 9 State Model, $J = h_0 + W_t\tau_f$, Controls and Hamiltonian

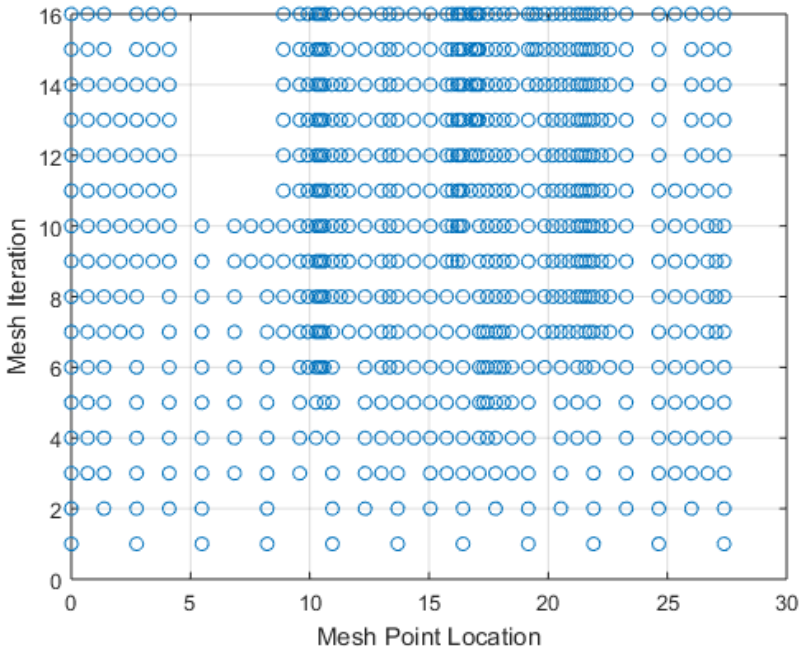


Figure A.6. High Hover, 9 State Model, $J = h_0 + W_t\tau_f$, Mesh History

A.1.3 11 State Model, Minimizing Altitude and Controls

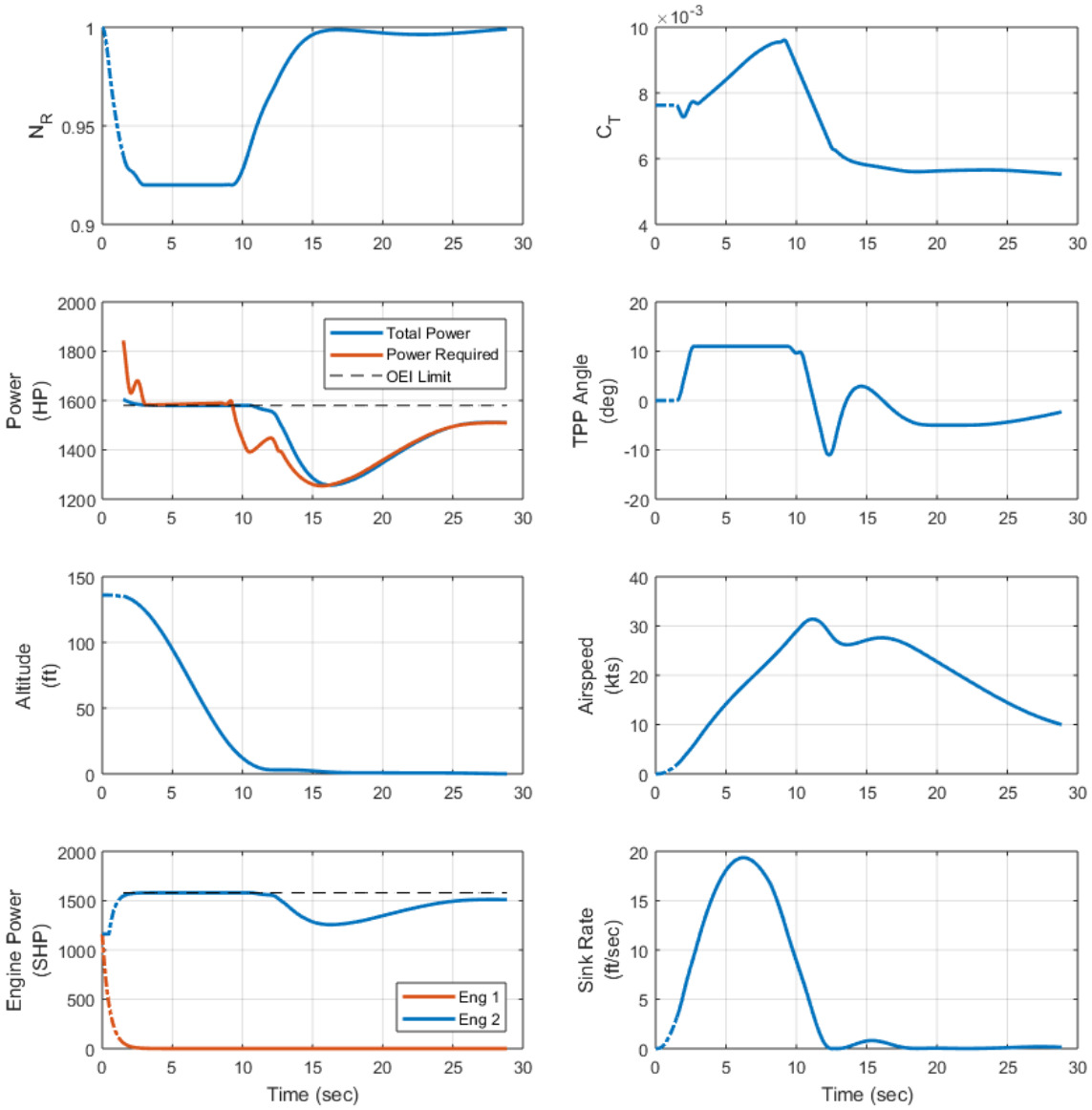
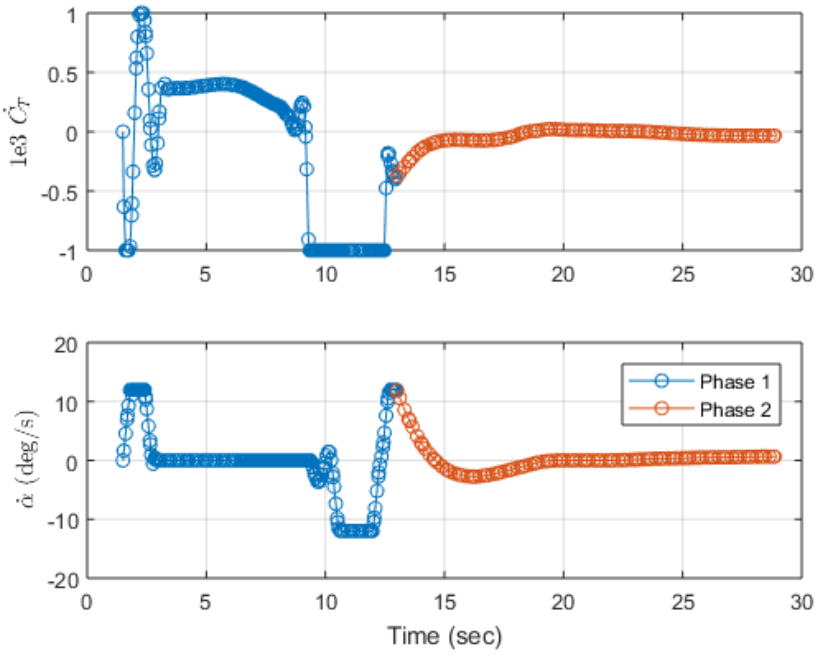
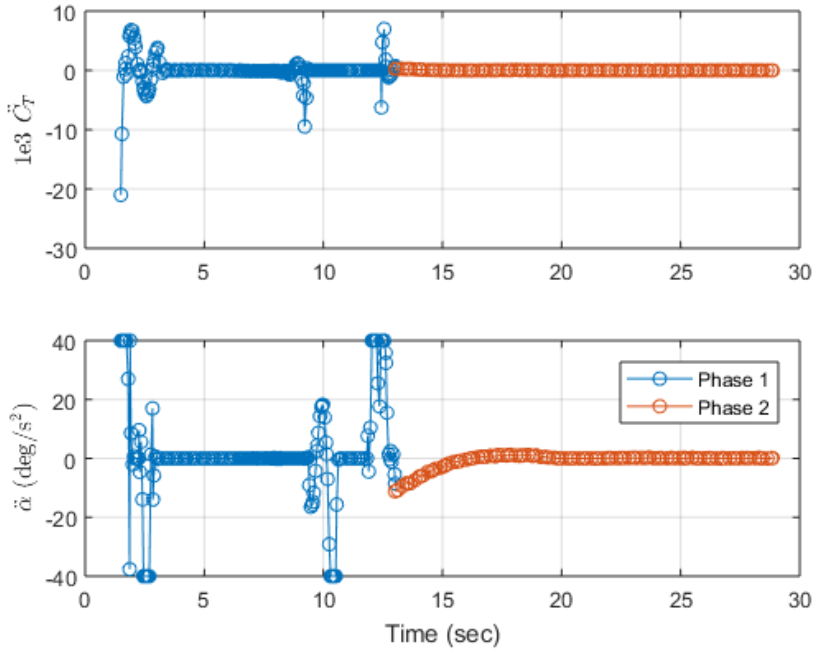


Figure A.7. High Hover, 11 State Model, $J = \pm h_0 + \int_{t_0}^{t_f} (W_1 u_1^2 + W_2 u_2^2) dt$

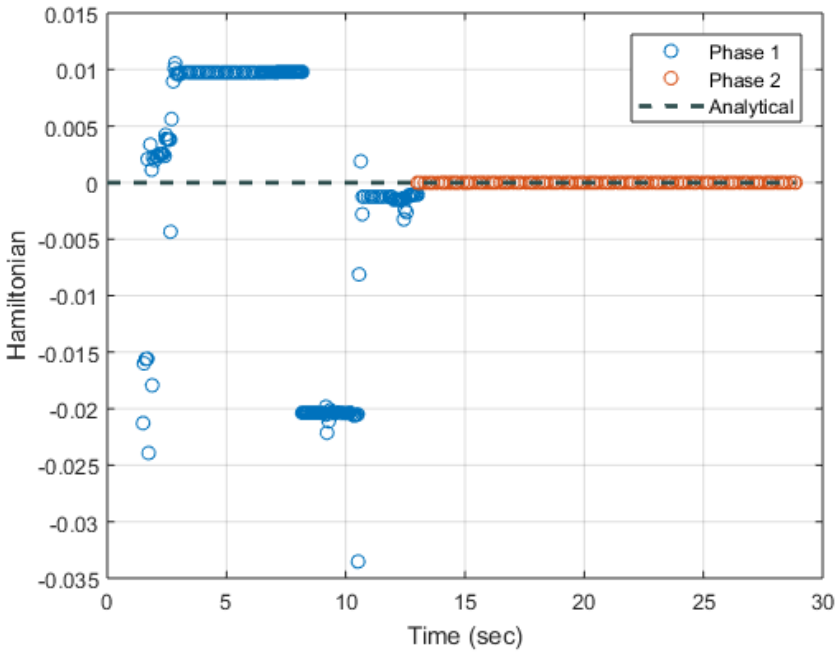


(a) C_T and α Rates

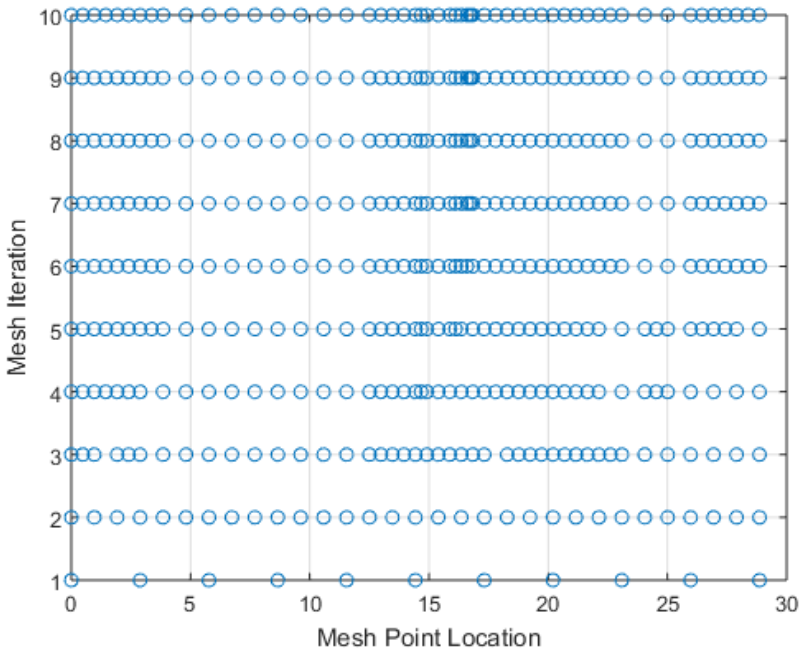


(b) Controls

Figure A.8. High Hover, 11 State Model, $J = \pm h_0 + \int_{t_0}^{t_f} \left(W_1 u_1^2 + W_2 u_2^2\right) dt$, Rates and Controls



(a) Hamiltonian



(b) Mesh History

Figure A.9. High Hover, 11 State Model, $J = \pm h_0 + \int_{t_0}^{t_f} \left(W_1 u_1^2 + W_2 u_2^2\right) dt$, Hamiltonian and Mesh History

A.2 10 Knot Point on Upper Portion of Curve

A.2.1 Nine State Model, Minimizing Altitude Only

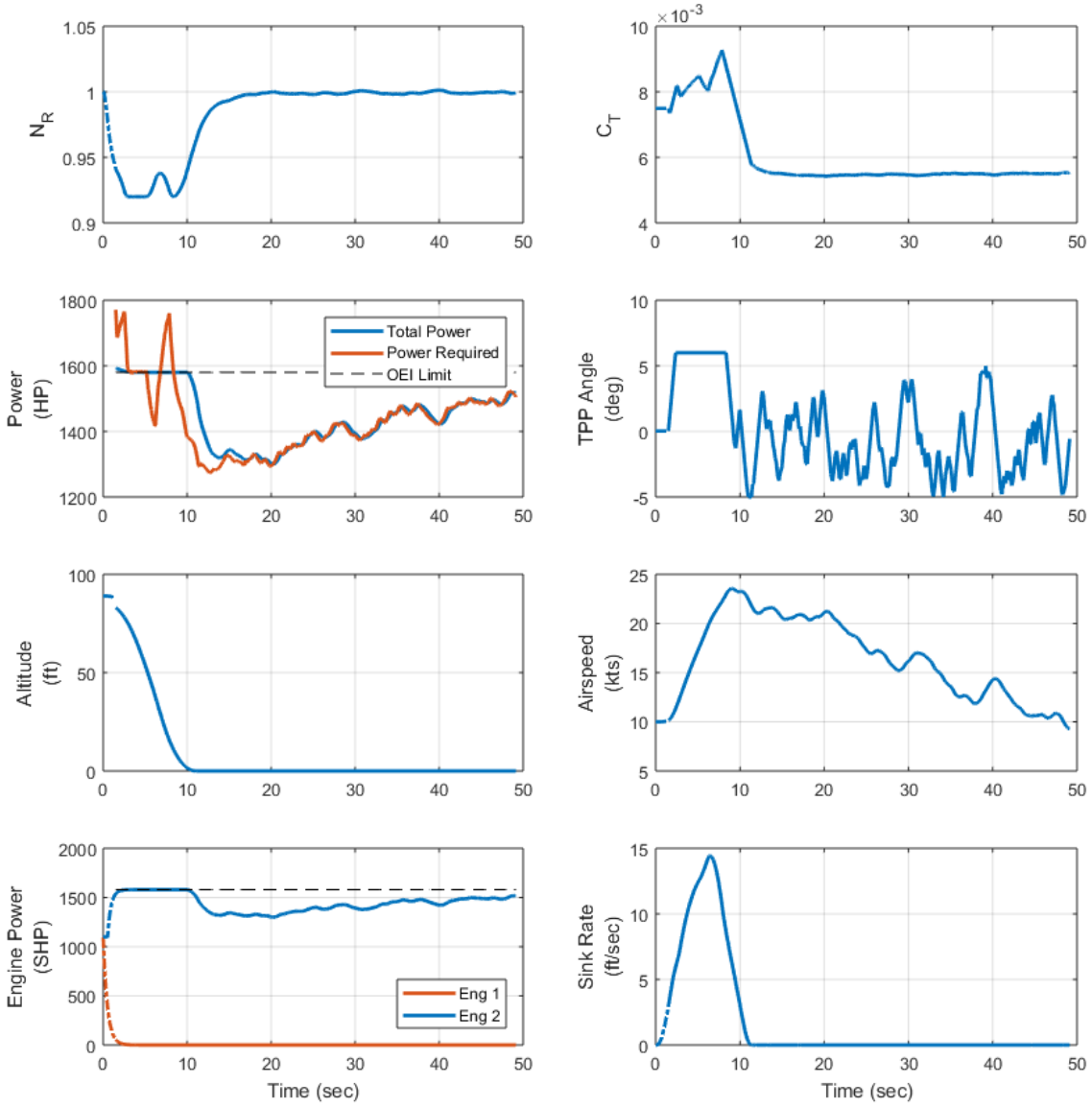
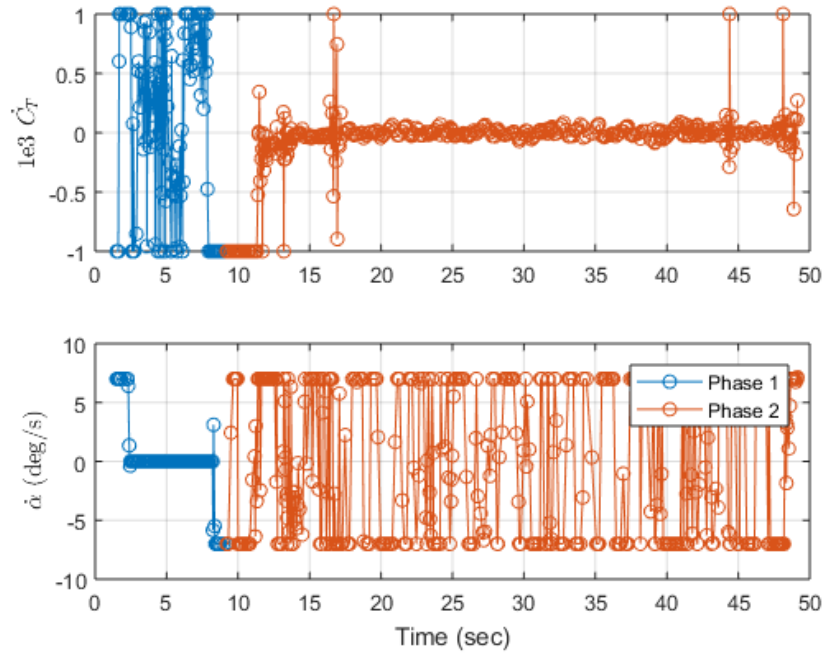
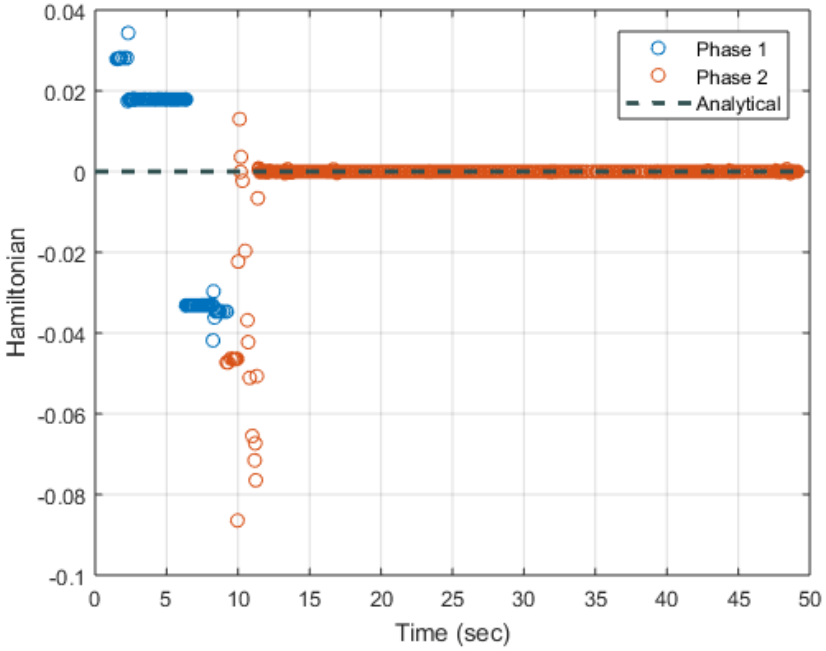


Figure A.10. 10 Knot Point, 9 State Model, $J = h_0$



(a) Controls



(b) Hamiltonian

Figure A.11. 10 Knot Point, 9 State Model, $J = h_0$, Controls and Hamiltonian

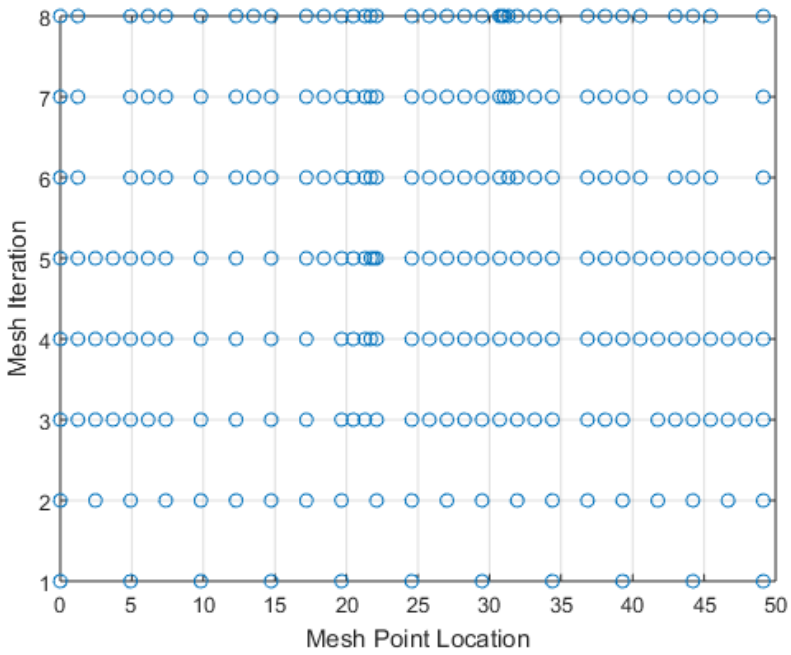


Figure A.12. 10 Knot Point, 9 State Model, $J = h_0$, Mesh History

A.2.2 Nine State Model, Minimizing Altitude and Final Time

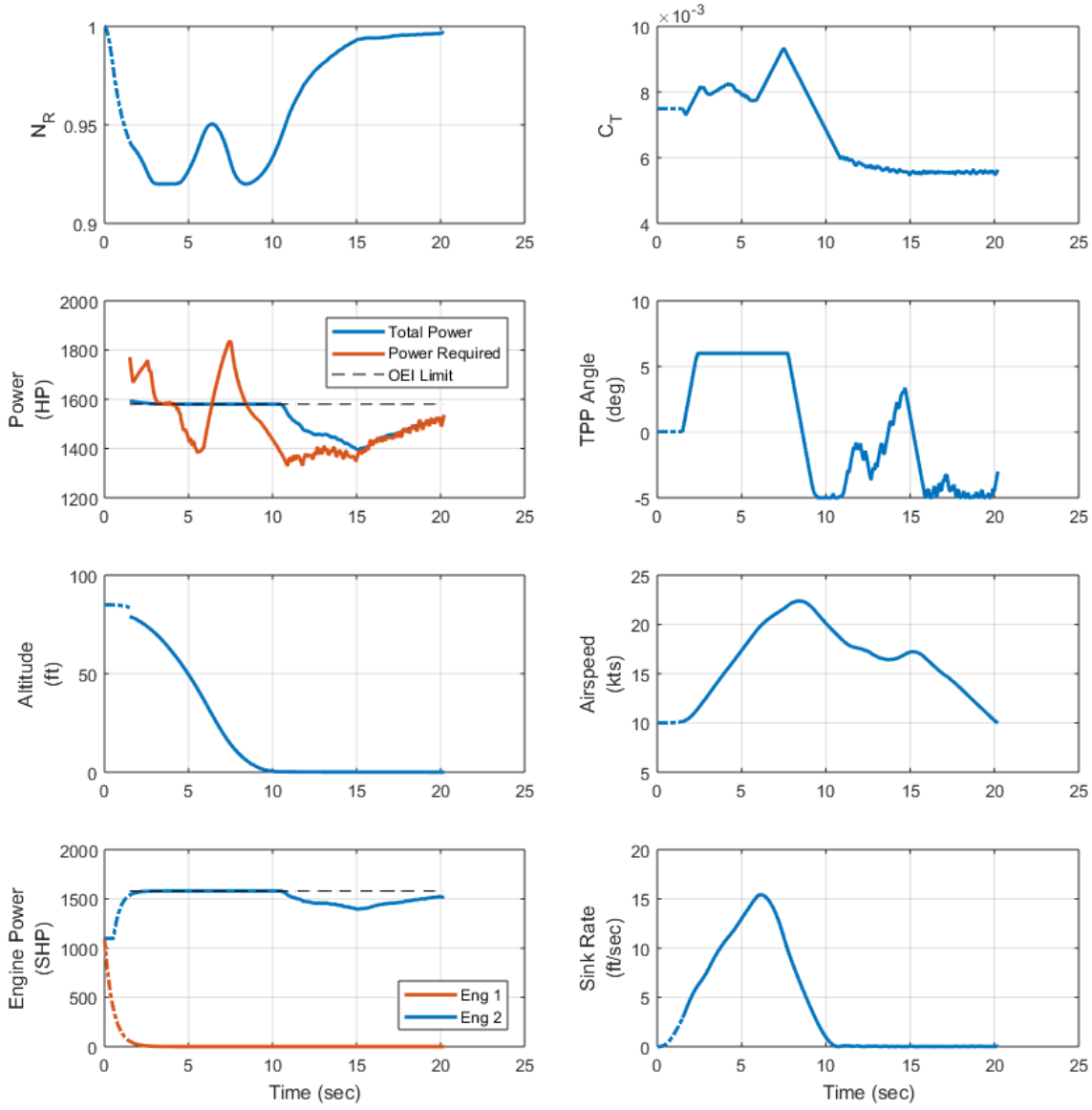
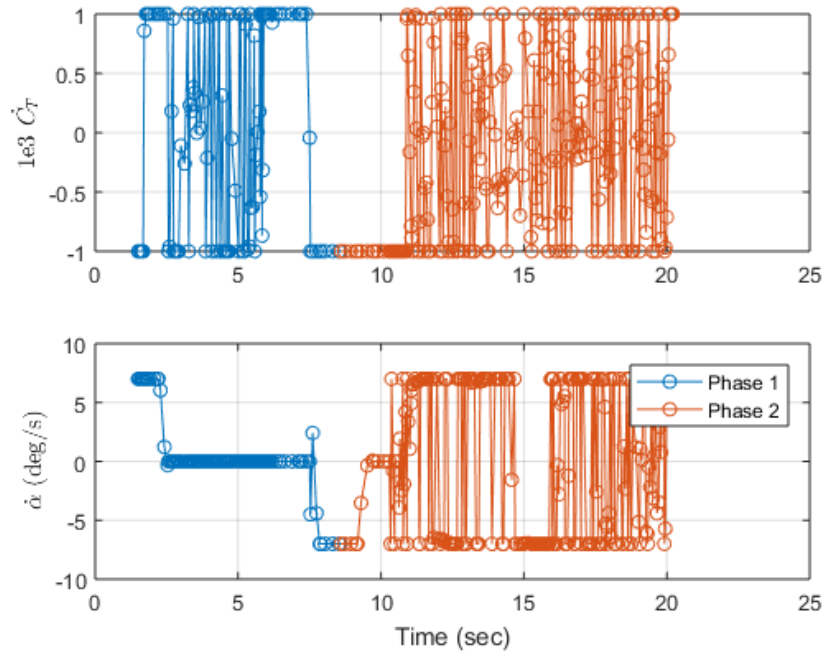
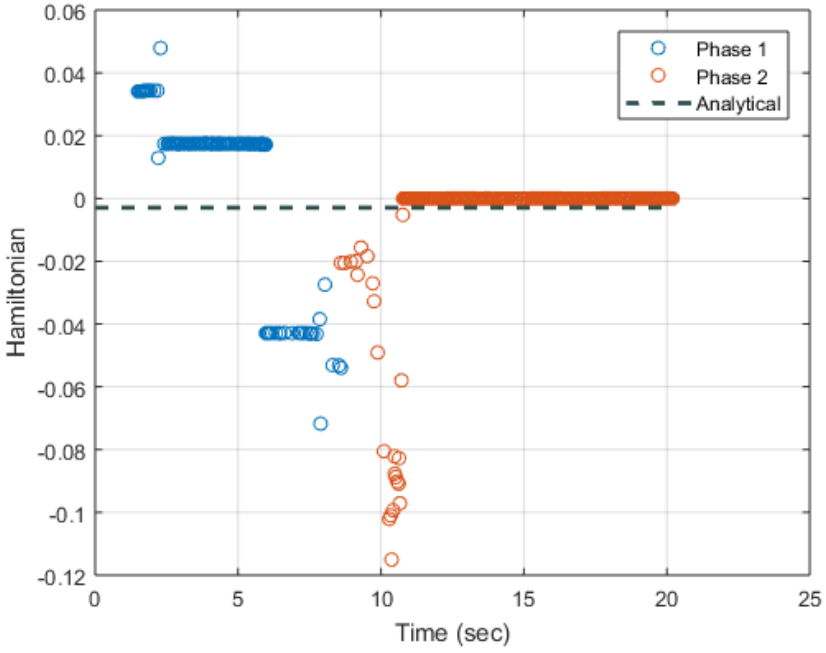


Figure A.13. 10 Knot Point, 9 State Model, $J = h_0 + W_t\tau_f$



(a) Controls



(b) Hamiltonian

Figure A.14. 10 Knot Point, 9 State Model, $J = h_0 + W_t\tau_f$, Controls and Hamiltonian

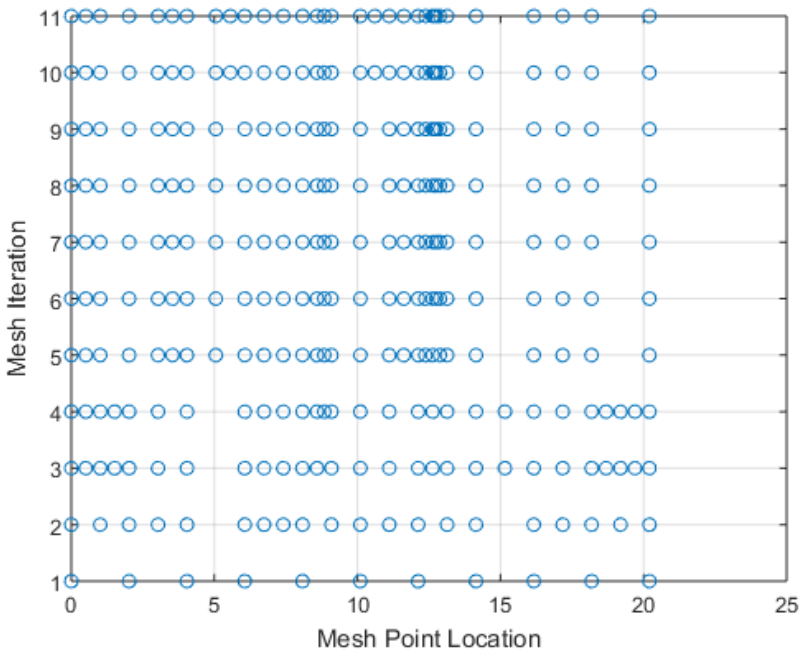


Figure A.15. 10 Knot Point, 9 State Model, $J = h_0 + W_t\tau_f$, Mesh History

A.2.3 11 State Model, Minimizing Altitude and Controls

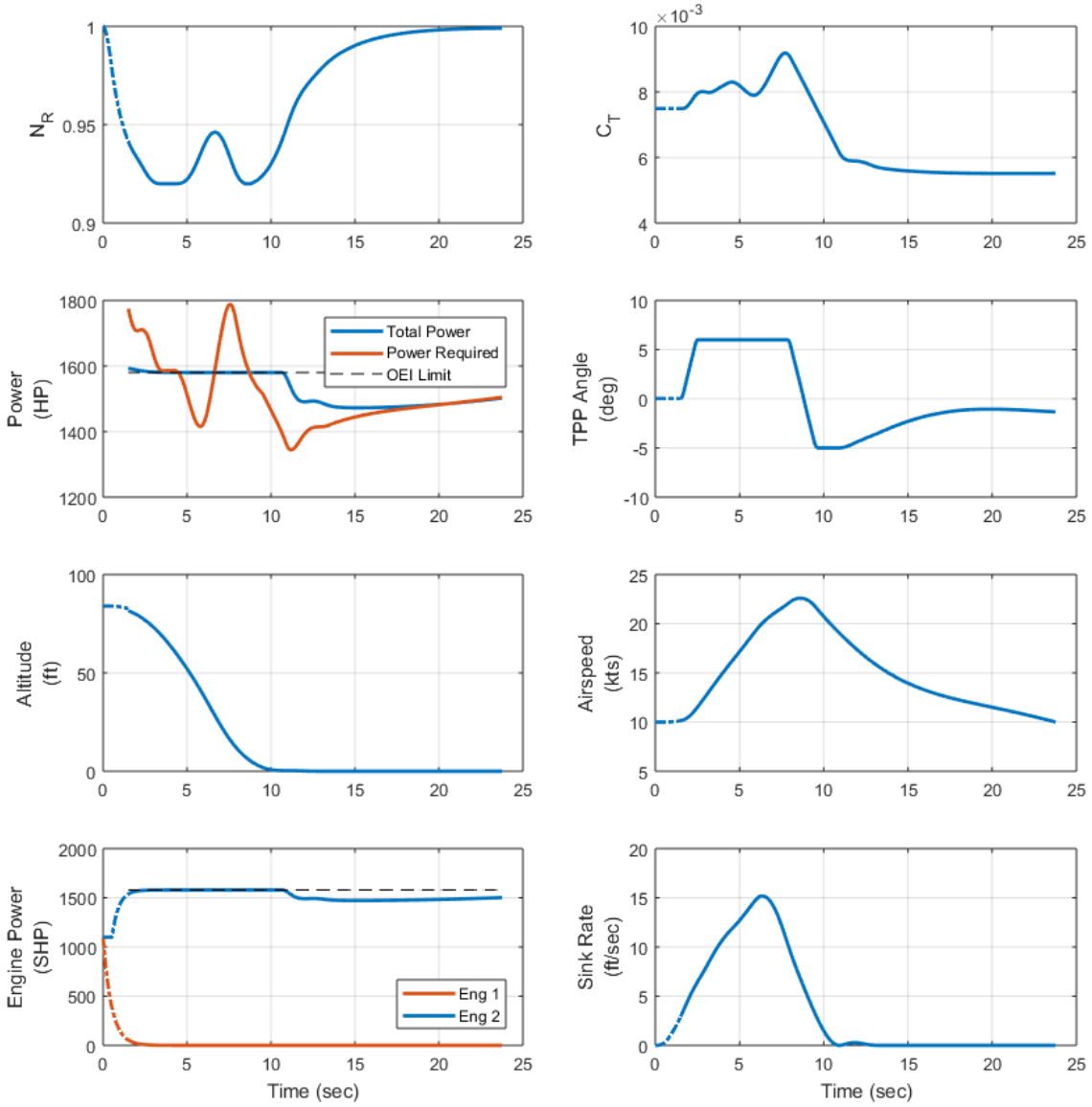
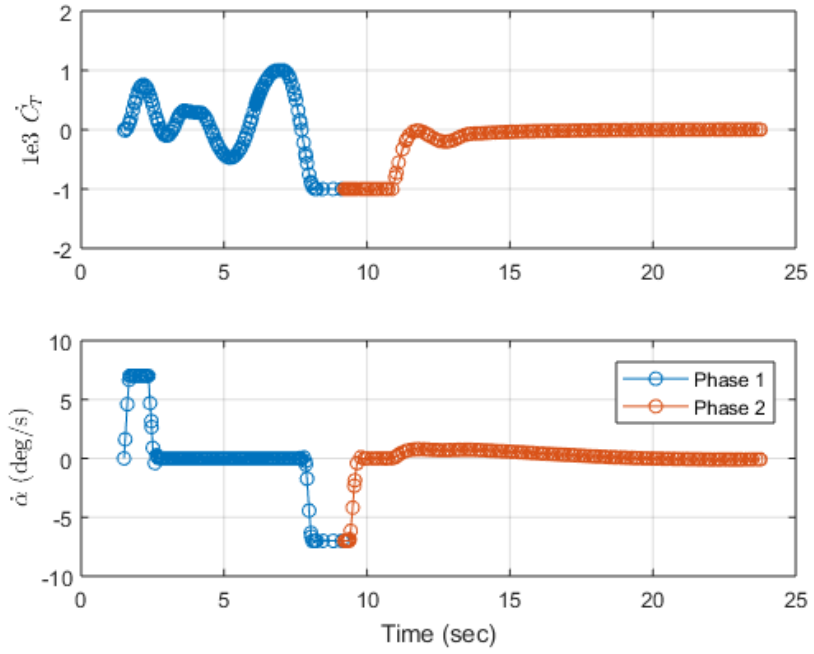
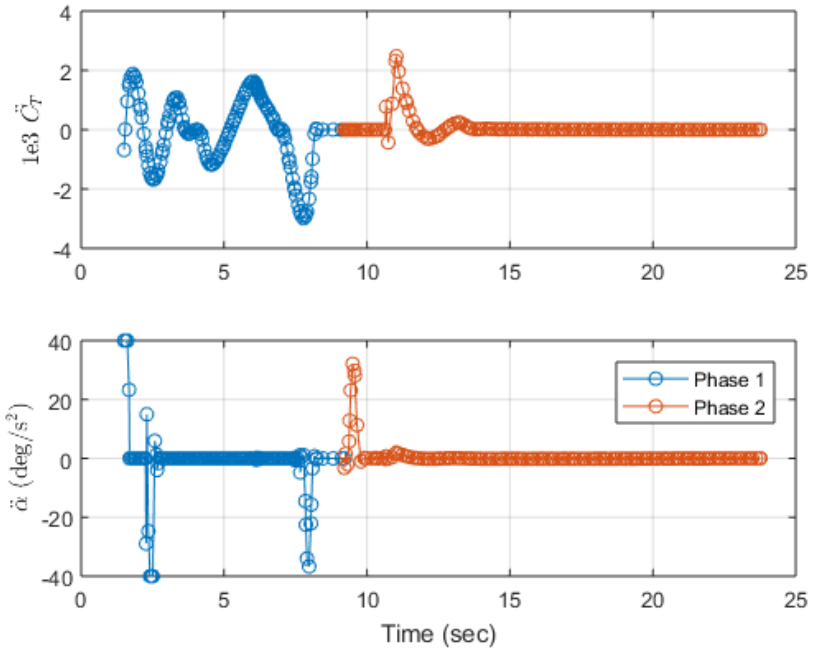


Figure A.16. 10 Knot Point, 11 State Model, $J = \pm h_0 + \int_{t_0}^{t_f} (W_1 u_1^2 + W_2 u_2^2) dt$

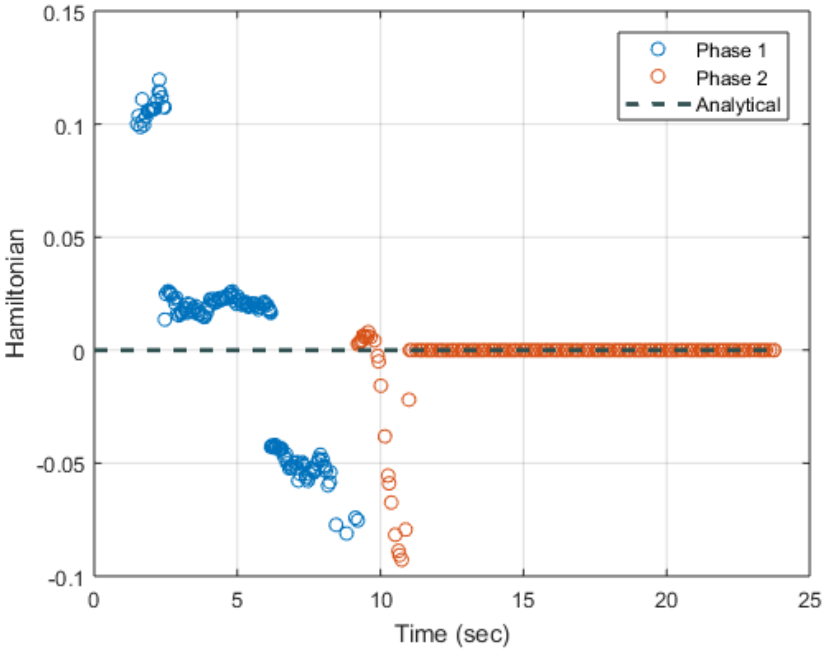


(a) C_T and α Rates

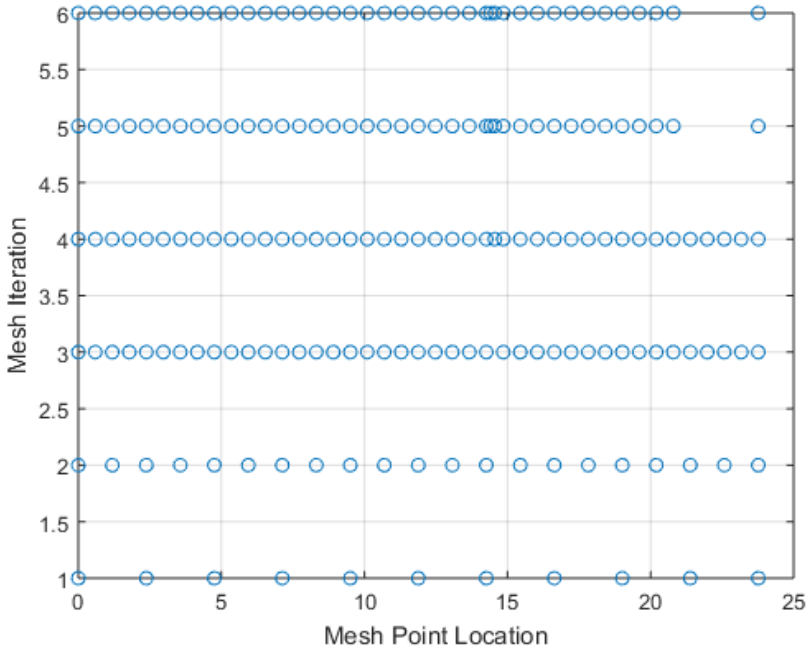


(b) Controls

Figure A.17. 10 Knot Point, 11 State Model, $J = \int_{t_0}^{t_f} \left(W_1 u_1^2 + W_2 u_2^2\right) dt$, Rates and Controls



(a) Hamiltonian



(b) Mesh History

Figure A.18. 10 Knot Point, 11 State Model, $J = \pm h_0 + \int_{t_0}^{t_f} \left(W_1 u_1^2 + W_2 u_2^2\right) dt$, Hamiltonian and Mesh History

A.3 Low Hover

A.3.1 Nine State Model, Minimizing Altitude Only

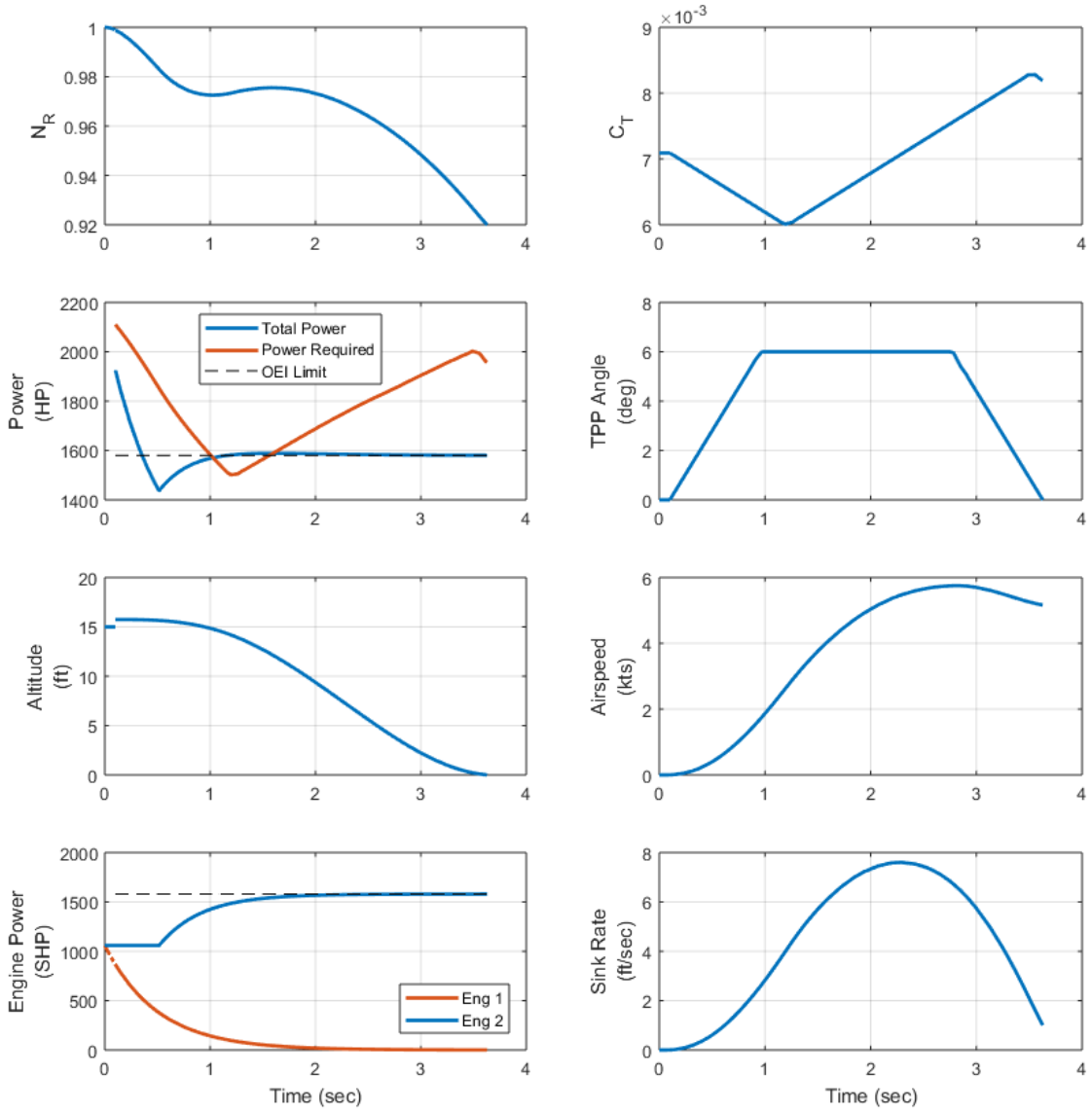
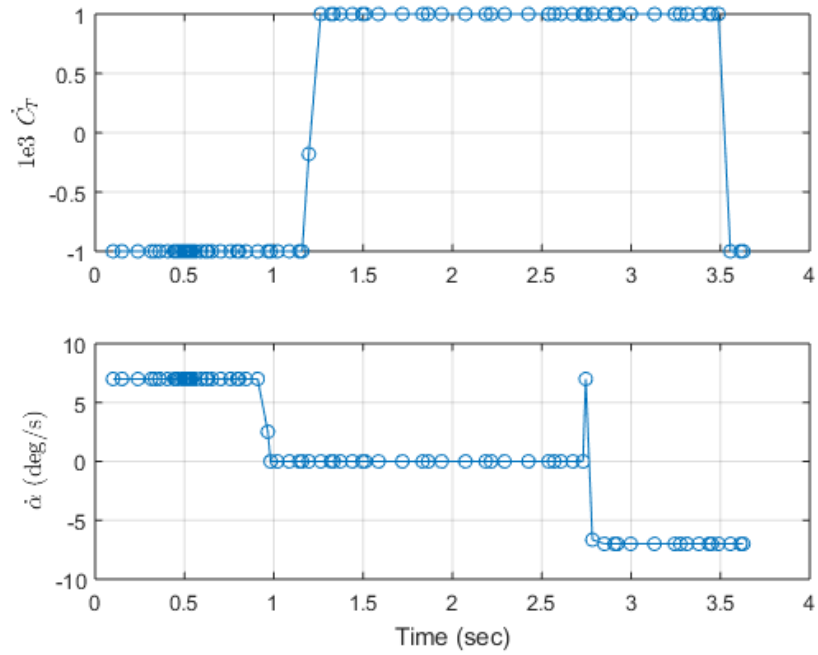
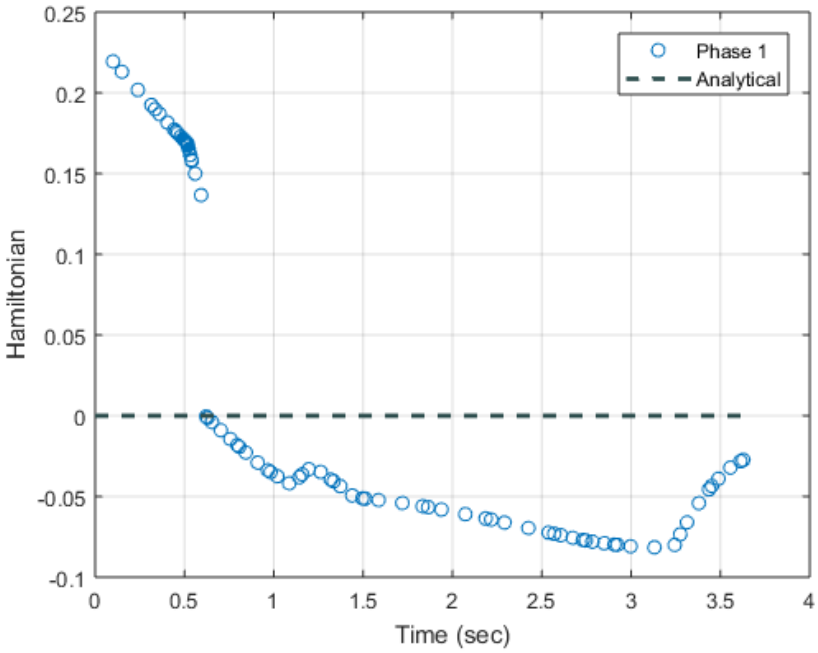


Figure A.19. Low Hover, 9 State Model, $J = h_0$



(a) Controls



(b) Hamiltonian

Figure A.20. Low Hover, 9 State Model, $J = h_0$, Controls and Hamiltonian

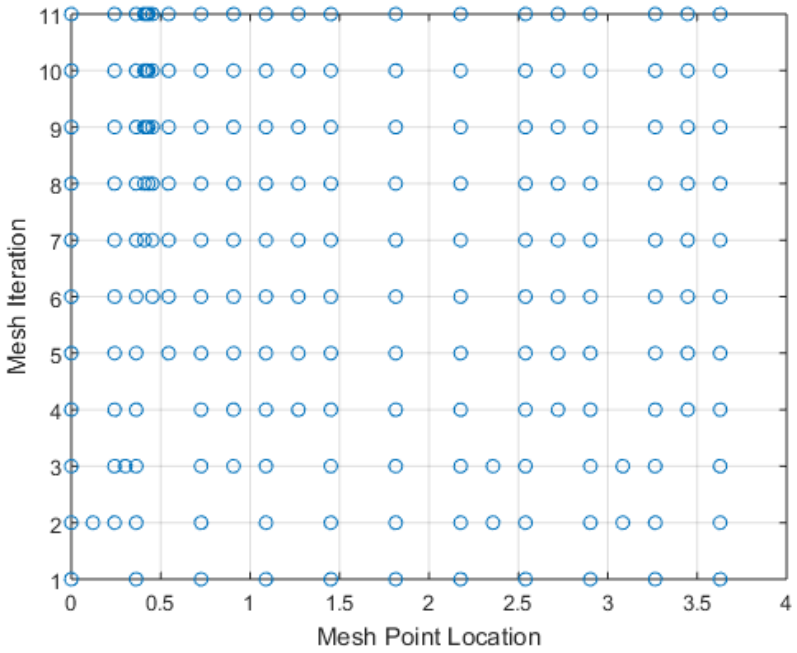


Figure A.21. Low Hover, 9 State Model, $J = h_0$, Mesh History

A.3.2 Nine State Model, Minimizing Altitude and Final Time

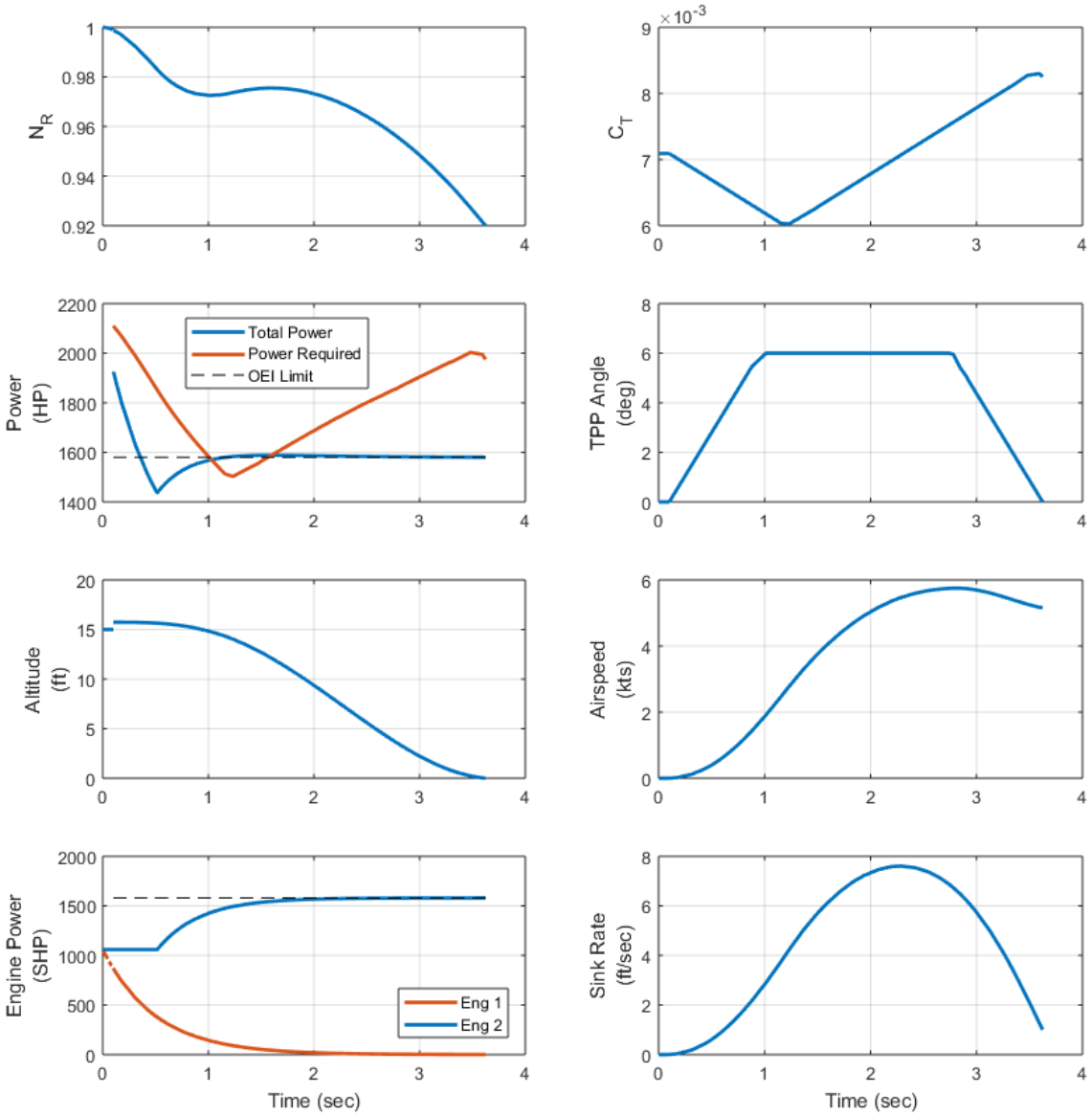
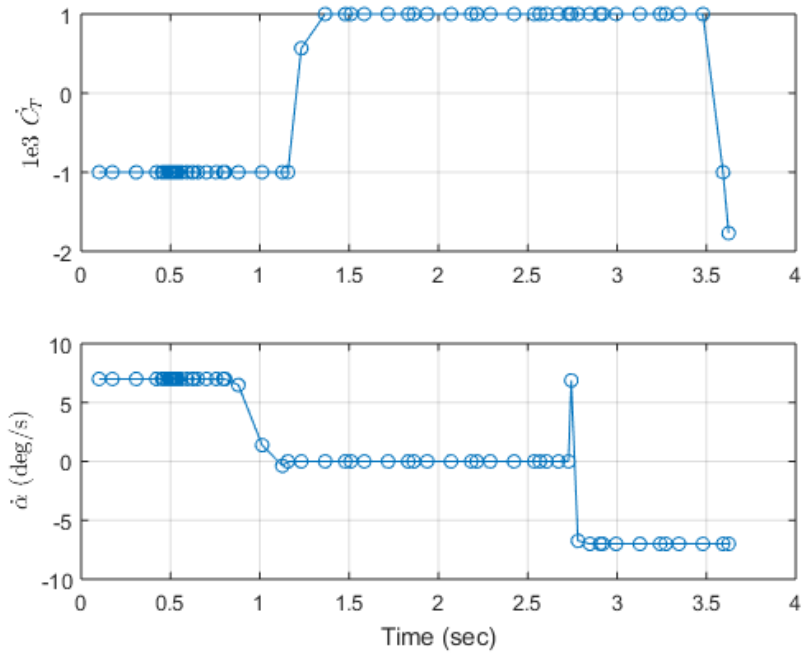
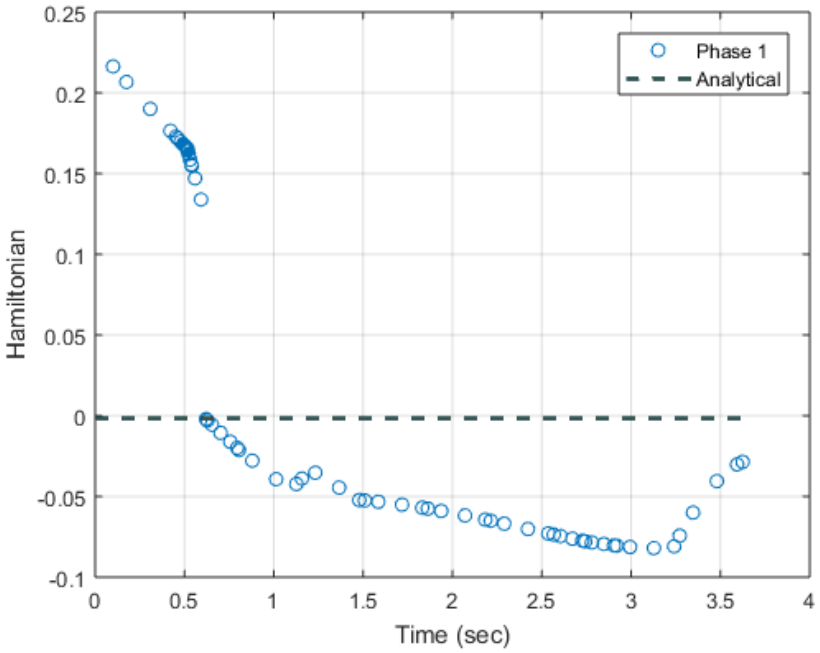


Figure A.22. Low Hover, 9 State Model, $J = h_0 + W_t\tau_f$



(a) Controls



(b) Hamiltonian

Figure A.23. Low Hover, 9 State Model, $J = h_0 + W_t\tau_f$, Controls and Hamiltonian

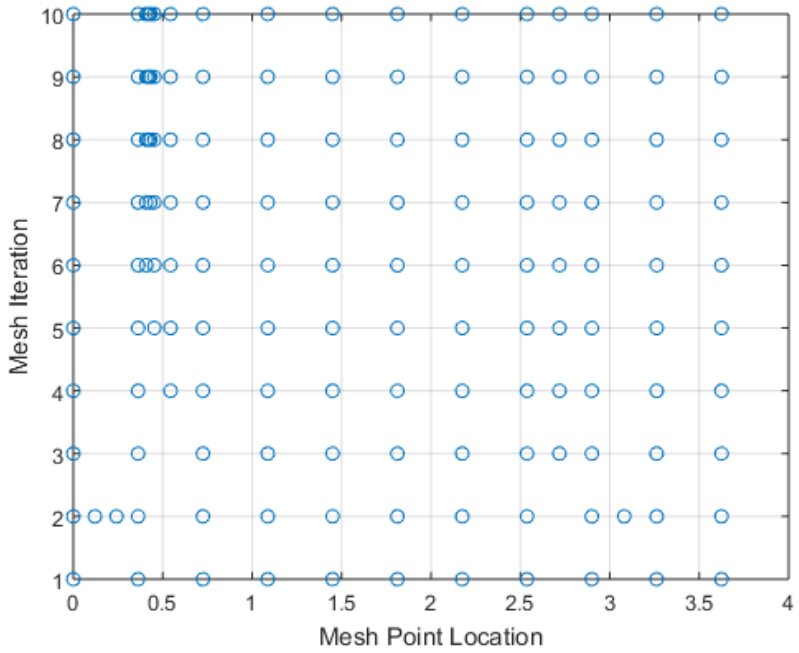


Figure A.24. Low Hover, 9 State Model, $J = h_0 + W_t\tau_f$, Mesh History

A.3.3 11 State Model, Minimizing Altitude and Controls

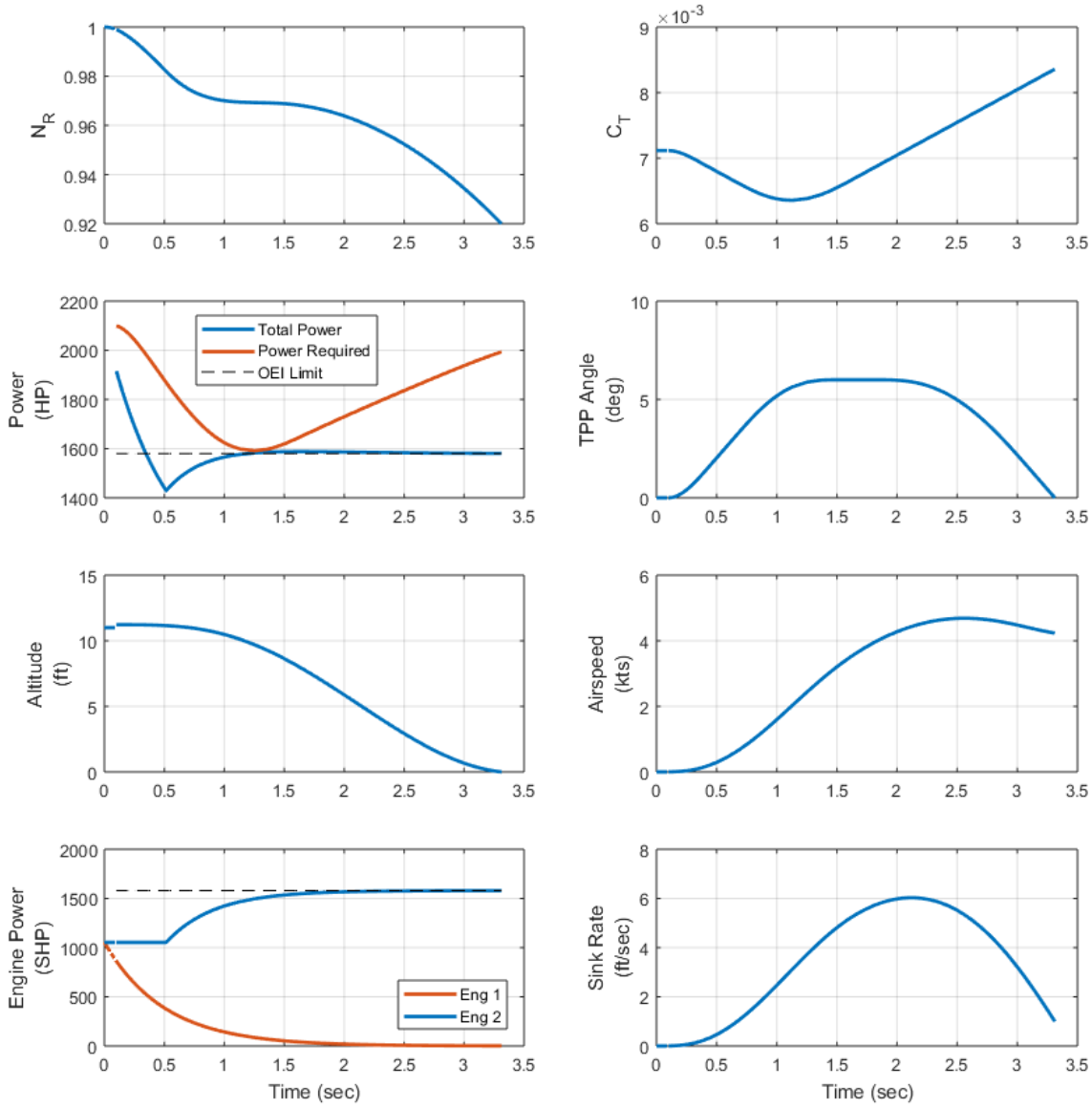
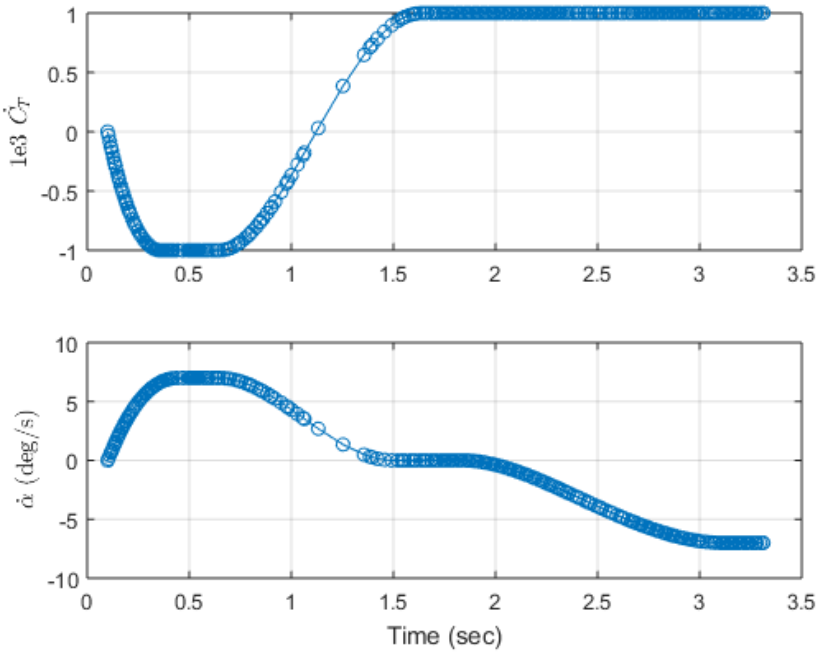
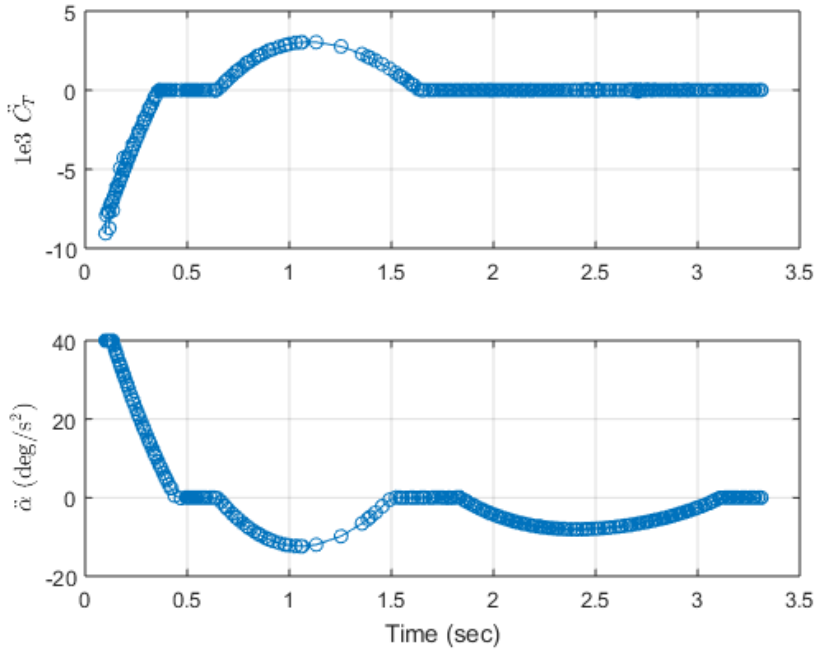


Figure A.25. Low Hover, 11 State Model, $J = \pm h_0 + \int_{t_0}^{t_f} \left(W_1 u_1^2 + W_2 u_2^2\right) dt$

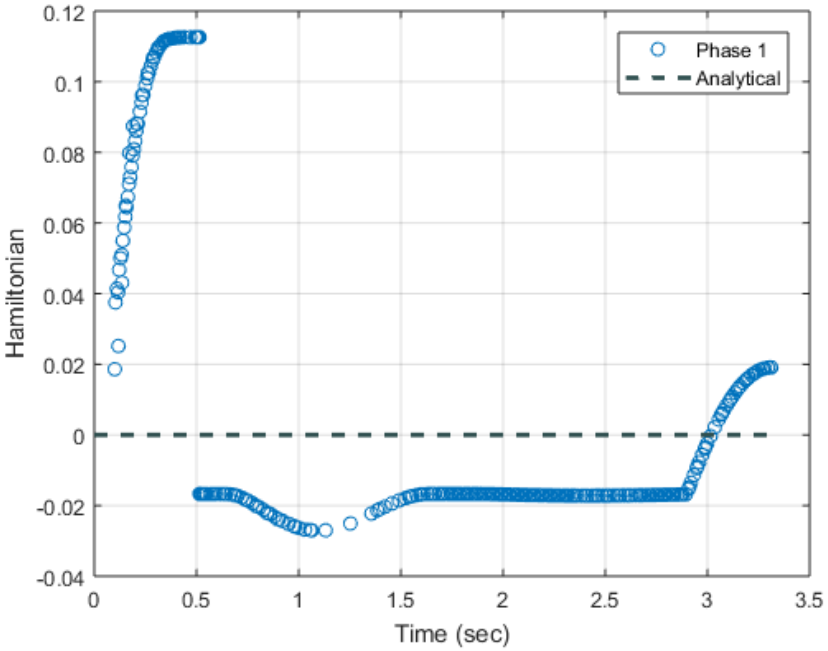


(a) C_T and α Rates

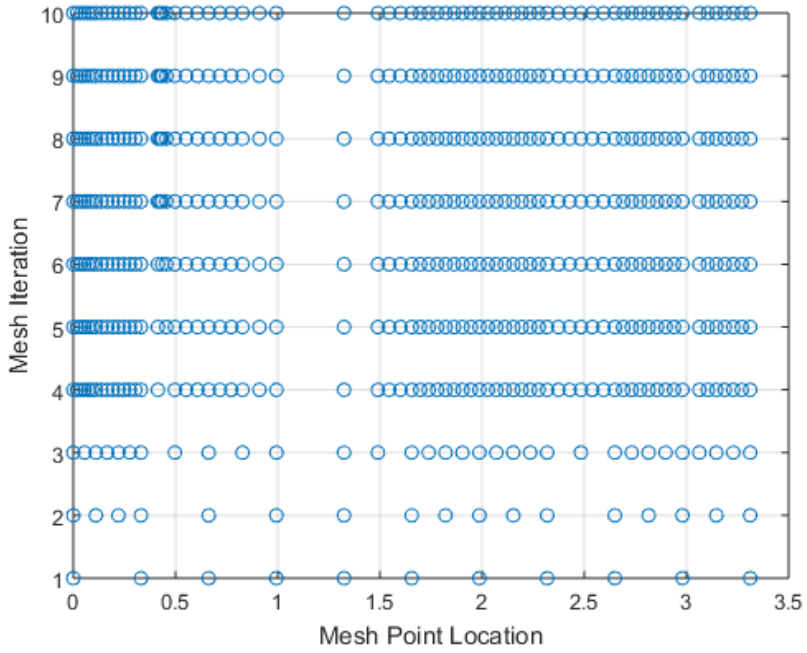


(b) Controls

Figure A.26. Low Hover, 11 State Model, $J = \pm h_0 + \int_{t_0}^{t_f} \left(W_1 u_1^2 + W_2 u_2^2\right) dt$, Rates and Controls



(a) Hamiltonian



(b) Mesh History

Figure A.27. Low Hover, 11 State Model, $J = \pm h_0 + \int_{t_0}^{t_f} \left(W_1 u_1^2 + W_2 u_2^2\right) dt$, Hamiltonian and Mesh History

Appendix B. Input Data and Results for Height Velocity Diagram Solutions

B.1 Nine State Model, Minimizing Altitude and Final Time

B.1.1 Upper Portion of Curve

Table B.1. HV Data and Inputs, 9 States, Upper Portion

Given u_0 (kts)	0	5	9	10	12	13
Solution h_0 (kts)	128	114	97	85	66	51
Solution t_f	28	19	22	20	19	17
Mesh Tolerance	$1x10^{-4}$	$1x10^{-4}$	$1x10^{-4}$	$1x10^{-4}$	$1x10^{-4}$	$1x10^{-4}$
NLP Tolerance	$1x10^{-6}$	$1x10^{-6}$	$1x10^{-6}$	$1x10^{-6}$	$1x10^{-6}$	$1x10^{-6}$
Derivative Stepsize	$1x10^{-8}$	$1x10^{-8}$	$1x10^{-8}$	$1x10^{-8}$	$1x10^{-8}$	$1x10^{-8}$
h_0 Guess (kts)	143	115	100	85	65	50
h_0 Lower Bound (kts)	40	40	40	40	40	40
h_0 Upper Bound (kts)	300	400	300	300	300	200
τ_f Guess	12	10	12	10	10	10
Max x_1 Guess	4	4	4	4	4	4
Max x_2 Guess	10	10	10	10	10	10
α Lower Bound (deg)	-11	-8	-6	-6	-6	-6
α Upper Bound (deg)	11	8	6	6	6	6
$\dot{\alpha}$ Lower Bound (deg/sec)	-12	-10	-7	-7	-7	-7
$\dot{\alpha}$ Upper Bound (deg/sec)	12	10	7	7	7	7

B.1.2 Lower Portion of Curve

Table B.2. HV Data and Inputs, 9 States, Lower Portion

Given u_0 (kts)	0	2	4	5
Solution h_0 (kts)	15	14	23	19
Solution t_f	3.6	5.3	5.4	5.8
Mesh Tolerance	$1x10^{-4}$	$1x10^{-4}$	$1x10^{-4}$	$1x10^{-4}$
NLP Tolerance	$1x10^{-8}$	$1x10^{-7}$	$1x10^{-6}$	$1x10^{-7}$
Derivative Stepsize	$1x10^{-9}$	$1x10^{-8}$	$1x10^{-8}$	$1x10^{-8}$
h_0 Guess (kts)	15	14	25	19
h_0 Lower Bound (kts)	1	1	1	1
h_0 Upper Bound (kts)	40	25	40	25
τ_f Guess	4	2	6	8
Max x_1 Guess	4	4	2	4
Max x_2 Guess	4	4	2	4
α Lower Bound (deg)	-5	-5	-10	-5
α Upper Bound (deg)	5	5	10	5
$\dot{\alpha}$ Lower Bound (deg/sec)	-5	-5	-5	-5
$\dot{\alpha}$ Upper Bound (deg/sec)	5	5	5	5

B.2 Eleven State Model, Minimizing Altitude and Controls

B.2.1 Upper Portion of Curve

Table B.3. HV Data and Inputs, 11 States, Upper Portion

Given u_0 (kts)	0	2.5	5	7	10	10.5
Solution h_0 (kts)	136	124	117	108	83	75
Solution t_f	29	25	21	18	17	48
Mesh Tolerance	$1x10^{-4}$	$1x10^{-4}$	$1x10^{-4}$	$1x10^{-4}$	$1x10^{-4}$	$1x10^{-4}$
NLP Tolerance	$1x10^{-6}$	$1x10^{-6}$	$1x10^{-6}$	$1x10^{-6}$	$1x10^{-6}$	$1x10^{-6}$
Derivative Stepsize	$1x10^{-8}$	$1x10^{-8}$	$1x10^{-8}$	$1x10^{-8}$	$1x10^{-8}$	$1x10^{-8}$
h_0 Guess (kts)	137	130	116	105	89	78
h_0 Lower Bound (kts)	30	40	30	40	40	40
h_0 Upper Bound (kts)	300	200	200	200	200	200
τ_f Guess	15	15	15	15	12	15
Max x_1 Guess	8	8	8	8	8	8
Max x_2 Guess	10	10	10	10	10	10
α Lower Bound (deg)	-11	-9.5	-8	-7	-6	-6
α Upper Bound (deg)	11	9.5	8	7	6	6
$\dot{\alpha}$ Lower Bound (deg/sec)	-12	-11	-10	-8	-7	-7
$\dot{\alpha}$ Upper Bound (deg/sec)	12	11	10	8	7	7
$\ddot{\alpha}$ Lower Bound (deg/sec^2)	-40	-40	-40	-40	-40	-40
$\ddot{\alpha}$ Upper Bound (deg/sec^2)	40	40	40	40	40	40

B.2.2 Lower Portion of Curve

Table B.4. HV Data and Inputs, 11 States, Lower Portion

Given u_0 (kts)	0	2	5	8	10	11	12
Solution h_0 (kts)	11	13	16	19	20	20	42
Solution t_f	3.3	3	5.1	4	4.5	4.1	4
Mesh Tolerance	$1x10^{-4}$	$1x10^{-4}$	$1x10^{-4}$	$1x10^{-4}$	$1x10^{-5}$	$1x10^{-5}$	$1x10^{-5}$
NLP Tolerance	$1x10^{-6}$	$1x10^{-6}$	$1x10^{-6}$	$1x10^{-6}$	$1x10^{-5}$	$1x10^{-5}$	$1x10^{-5}$
Derivative Stepsize	$1x10^{-8}$	$1x10^{-8}$	$1x10^{-8}$	$1x10^{-8}$	$1x10^{-9}$	$1x10^{-8}$	$1x10^{-8}$
h_0 Guess (kts)	9	12	16	19	20	20	35
h_0 Lower Bound (kts)	1	1	1	1	1	1	10
h_0 Upper Bound (kts)	40	40	40	30	40	40	80
τ_f Guess	4	4	4	4	10	10	10
Max x_1 Guess	2	2	2	2	4	4	4
Max x_2 Guess	2	2	2	2	4	4	4
α Lower Bound (deg)	-6	-6	-6	-6	-5	-6	-4
α Upper Bound (deg)	6	6	6	6	5	6	4
$\dot{\alpha}$ Lower Bound (deg/sec)	-7	-7	-7	-7	-5	-7	-4
$\dot{\alpha}$ Upper Bound (deg/sec)	7	7	7	7	5	7	4
$\ddot{\alpha}$ Lower Bound (deg/sec^2)	-15	-15	-15	-15	-15	-20	-15
$\ddot{\alpha}$ Upper Bound (deg/sec^2)	15	15	15	15	15	20	15

Appendix C. Solution Guess and Convergence Study

C.1 Solution Guess

Three values were assigned to each state and control for the solution guess. The resulting three column matrix is depicted below for the indicated problem segment.

Table C.1. Phase 1 Guess for Upper Portion of Curve

Variable	1^{st} Value	2^{nd} Value	3^{rd} Value
τ	0	$0.5\tau_{fguess}$	τ_{fguess}
x_1	x_{1_1}	x_{1maxg}	0
x_2	x_{2_1}	x_{2maxg}	2
x_3	x_{3_1}	0.98	0.95
x_4	x_{4_0}	$0.5x_{4_1}$	$0.25x_{4_1}$
x_5	x_{5_1}	1	3
x_6	x_{6_1}	0	0
x_7	x_{7_1}	1	1
x_8	x_{8_1}	x_{8_1}	x_{8_1}
x_9	x_{9_1}	x_{9_1}	x_{9_1}
x_{10}	0	0	0
x_{11}	0	0	0
u_1	1	1	1
u_2	0	0	0

Table C.2. Phase 2 Guess for Upper Portion of Curve

Variable	1^{st} Value	2^{nd} Value	3^{rd} Value
τ	$\tau_{fguess} + 0.1$	$\tau_{fguess} + 0.5$	$\tau_{fguess} + 1$
x_1	0	1	0
x_2	2	1	0
x_3	0.96	0.95	0.92
x_4	$0.2x_{4_1}$	0.05	0
x_5	3.1	3.5	4
x_6	0	0	0
x_7	0.9	1	1
x_8	x_{8_1}	x_{8_1}	x_{8_1}
x_9	x_{9lop2}	x_{9lop2}	x_{9_f}
x_{10}	0	0	0
x_{11}	0	0	0
u_1	1	1	1
u_2	0	0	0

Table C.3. Guess for Lower Portion of Curve

Variable	1^{st} Value	2^{nd} Value	3^{rd} Value
τ	0	$0.5\tau_{fguess}$	τ_{fguess}
x_1	x_{1_1}	x_{1maxg}	0
x_2	x_{2_1}	x_{2maxg}	2
x_3	x_{3_1}	0.98	0.95
x_4	x_{4_0}	$0.5x_{4_1}$	0
x_5	x_{5_1}	1	3
x_6	x_{6_1}	0	0
x_7	x_{7_1}	1	1
x_8	x_{8_1}	x_{8_1}	x_{8_1}
x_9	x_{9_1}	x_{9_1}	x_{9_1}
x_{10}	0	0	0
x_{11}	0	0	0
u_1	1	1	1
u_2	1	1	1

C.2 Convergence Study

A convergence study was conducted to determine sufficient values for the mesh and nonlinear programming (NLP) tolerances. Note that the bounds on pitch angle were different during this study than those used in the end solution, yielding a different initial altitude from results in Chapter 4.

Table C.4. Convergence Study Part 1

Mesh Tolerance	NLP Tolerance	h_0	Hamiltonian RMS Error
$1x10^{-1}$	$1x10^{-1}$	173	$6.40x10^{8}$
$1x10^{-2}$	$1x10^{-1}$	169	$6.60x10^{8}$
$1x10^{-3}$	$1x10^{-1}$	169	$1.19x10^{8}$
$1x10^{-1}$	$1x10^{-2}$	176	$2.67x10^{8}$
$1x10^{-2}$	$1x10^{-2}$	171	0.68
$1x10^{-3}$	$1x10^{-2}$	168	0.04
$1x10^{-4}$	$1x10^{-2}$	144	0.04
$1x10^{-1}$	$1x10^{-3}$	171	4.65
$1x10^{-2}$	$1x10^{-3}$	171	4.65
$1x10^{-3}$	$1x10^{-3}$	168	0.41
$1x10^{-4}$	$1x10^{-3}$	161	0.60
$1x10^{-1}$	$1x10^{-4}$	164	0.31
$1x10^{-2}$	$1x10^{-4}$	158	0.78
$1x10^{-3}$	$1x10^{-4}$	157	1.20
$1x10^{-4}$	$1x10^{-4}$	130	0.01
$1x10^{-5}$	$1x10^{-4}$	130	0.01
$1x10^{-6}$	$1x10^{-4}$	130	0.01

Table C.5. Convergence Study Part 2

Mesh Tolerance	NLP Tolerance	h_0	Hamiltonian RMS Error
$1x10^{-1}$	$1x10^{-5}$	164	0.31
$1x10^{-2}$	$1x10^{-5}$	162	0.24
$1x10^{-3}$	$1x10^{-5}$	156	0.77
$1x10^{-4}$	$1x10^{-5}$	130	0.01
$1x10^{-5}$	$1x10^{-5}$	130	0.01
$1x10^{-6}$	$1x10^{-5}$	130	0.01
$1x10^{-1}$	$1x10^{-6}$	160	0.06
$1x10^{-2}$	$1x10^{-6}$	129	0.01
$1x10^{-3}$	$1x10^{-6}$	129	0.02
$1x10^{-4}$	$1x10^{-6}$	129	0.01
$1x10^{-5}$	$1x10^{-6}$	129	0.01
$1x10^{-1}$	$1x10^{-7}$	130	0.01
$1x10^{-2}$	$1x10^{-7}$	130	0.01
$1x10^{-3}$	$1x10^{-7}$	130	0.01
$1x10^{-4}$	$1x10^{-7}$	130	0.01
$1x10^{-5}$	$1x10^{-7}$	130	0.01
$1x10^{-1}$	$1x10^{-8}$	132	0.01
$1x10^{-2}$	$1x10^{-8}$	131	0.01
$1x10^{-3}$	$1x10^{-8}$	129	0.02
$1x10^{-4}$	$1x10^{-8}$	129	0.01
$1x10^{-1}$	$1x10^{-9}$	131	0.01
$1x10^{-2}$	$1x10^{-9}$	130	0.01
$1x10^{-4}$	$1x10^{-9}$	130	0.02
$1x10^{-5}$	$1x10^{-9}$	130	0.01

Appendix D. MATLAB Code

D.1 Code Utilized During Research

This appendix contains the MATLAB®m files and functions used to compute the Height-Velocity (HV) diagram. In the GPOPS-II® main file, two functions were called to determine initial conditions prior to engine failure and to integrate the dynamics equations to mimic pilot delay. These two functions are listed first. The next series of function files was the method used to calculate induced velocity, taking into account vortex ring state, from Johnson's work in[23]. Finally, the GPOPS-II® main file is listed along with the continuous function and endpoint function which are needed to run the main file.

D.2 Initial Condition Function

```
%-------------------------------------------------%
%---------- Begin Function: AH1Init ------------ %
%-------------------------------------------------%
function dxp = AH1init_v2o(x,x0,const)

% Read in known initial conditions
x1 = x0(1); x2 = x0(2); x3 = x0(3); x4 = x0(4);

% This is what the function solves:
x7 = x(1); x8 = x(2); x9 = x(3);
x6 = x7;

% Assign auxdata to structure "const"
useVRS  = const.useVRS;
g0      = const.g0;
m0      = const.m0;
f0z     = const.f0z;
f0x     = const.f0x;
i0      = const.i0;
M       = const.M;
Mtr     = const.Mtr;
etaMR   = const.etaMR;
etaTR   = const.etaTR;
etaCBOX = const.etaCBOX;
HPacc   = const.HPacc;
p1      = const.p1;
p2      = const.p2;
p3      = const.p3;
p4      = const.p4;
pt      = const.pt;
```

```
Poei    = const.Poei;
R       = const.R;
hh      = const.hh;
fv      = const.fv ;
x2w     = const.x2w;
ki      = const.ki;

%---- Calculate Ground Effect --------------------%
z = max(hh+10.*R.*x4,hh);
% Hayden
kg = (0.9926 + 0.03794.*(z./(2*R)).^-2).^(2/3);
kg(z>(2*R)) = 1;

%---- Calculate Washout --------------------------%
fw = 1-1/x2w*abs(x2);
fw(x2>x2w) = 0;

%---- Aerodynamic Model---------------------------%
% Calculate Johnson's mux = Vx_rotor/vh, muz = -Vz_rotor/vh
% updated with revised equations for mu
mux = 0.4472./(x3.*sqrt(x8)).*(x2.*cos(x9) + x1.*sin(x9));
muz = 0.4472./(x3.*sqrt(x8)).*(x2.*sin(x9) - x1.*cos(x9));

lam_i_J = ones(length(x1),1);

for m = 1:length(x1)
% Call getInducedVelocityVRS.m to obtain lambda_i
lam_i_J(m) = getInducedVelocityVRS3(mux(m),muz(m),useVRS);
end

% Regular mu, and lambda_i
mu  = 0.01./x3.*(x2.*cos(x9) + x1.*sin(x9));
lam_i = 0.0224.*lam_i_J.*(x8).^.5;

%---- End Aerodynamic Model-----------------------%

%---- Calculate Power Requred --------------------%

% Main Rotor
Cp_mr = ki.*lam_i.*x8./1000 + p1.*(1+4.65.*mu.^2) + p2.*mu.^3 - p3.*x1./x3.^3;
Pr_mr = 1/M.*p4.*Cp_mr.*x3.^3;

% Tail Rotor
Pr_tr = (1-1.2*mu)*1./Mtr.*sqrt(Pr_mr.^3./(pt.*x3.^3));

% Total P_req including accessories and losses
Pr = (Pr_mr/etaMR+Pr_tr/etaTR+HPacc*550)/etaCBOX;
pr1 = Pr./Poei;

%---- Calculate Derivatives ----------------------%
dx1p = g0 - m0.*x3.^2.*x8.*(kg-fv.*fw).*cos(x9) - ...
       f0z.*x1.*sqrt(x1.^2+x2.^2);
dx2p = m0.*x3.^2.*x8.*(kg-fv.*fw).*sin(x9) - ...
       f0x.*x2.*sqrt(x1.^2+x2.^2);
dx3p = i0./x3.*(x6+x7-pr1);

dxp = [dx1p, dx2p, dx3p]';
```

D.3 Pilot Delay Function

```
%-------------------------------------------------%
% ------- Begin Function:  AH-1Z Dynamics ------- %
%-------------------------------------------------%
function dxp = ah1zdelay13(t,x,u1,u2,const)

x1 = x(1); x2 = x(2); x3 = x(3); x4 = x(4); %x5 = x(5);
x6 = x(6); x7 = x(7); x8 = x(8); x9 = x(9);
x10 = x(10); x11 = x(11);


%--- Retrieve Constants from structure -----------%
useVRS  = const.useVRS;
g0      = const.g0;
m0      = const.m0;
f0z     = const.f0z;
f0x     = const.f0x;
i0      = const.i0;
k1      = const.k1;
k2      = const.k2;
G1      = const.G1;
M       = const.M;
Mtr     = const.Mtr;
etaMR   = const.etaMR;
etaTR   = const.etaTR;
etaCBOX = const.etaCBOX;
HPacc   = const.HPacc;
ki      = const.ki;
p1      = const.p1;
p2      = const.p2;
p3      = const.p3;
p4      = const.p4;
pt      = const.pt;
Poei    = const.Poei;
% loss    = const.loss;
R       = const.R;
hh      = const.hh;
fv      = const.fv ;
x2w     = const.x2w;
tref    = const.tref;

%---- Calculate Ground Effect --------------------%
z = max(hh+10.*R.*x4,hh);
% Hayden
kg = min((0.9926 + 0.03794.*(z./(2*R)).^-2).^(2/3),1.3);
kg(z>(2*R)) = 1;

%---- Calculate Washout --------------------------%
fw = 1-1/x2w.*abs(x2);
fw(x2>x2w) = 0;

%---- Aerodynamic Model---------------------------%
% Calculate Johnson's mux = Vx_rotor/vh, muz = -Vz_rotor/vh
% updated with revised equations for mu
mux = 0.4472./(x3.*sqrt(x8)).*(x2.*cos(x9) + x1.*sin(x9));
muz = 0.4472./(x3.*sqrt(x8)).*(x2.*sin(x9) - x1.*cos(x9));

lam_i_J = ones(length(x1),1);

for m = 1:length(x1)
% Call getInducedVelocityVRS.m to obtain lambda_i
lam_i_J(m) = getInducedVelocityVRS3(mux(m),muz(m),useVRS);
end

% Regular mu, and lambda_i
```

```
mu  = 0.01./x3.*(x2.*cos(x9) + x1.*sin(x9));
lam_i = 0.0224.*lam_i_J.*(x8).^.5;
% lam = 0.01./x3.*(x2.*sin(x9) - x1.*cos(x9)) + 0.0224.*lam_i.*x8.^.5;

%---- End Aerodynamic Model-----------------------%


%---- Calculate Power Requred -------------------%
% Main Rotor
Cp_mr = ki.*lam_i.*x8./1000 + p1.*(1+4.65.*mu.^2) + p2.*mu.^3 - p3.*x1./x3.^3;
Pr_mr = 1/M.*p4.*Cp_mr.*x3.^3;
% Tail Rotor
Pr_tr = (1-1.2*mu)*1./Mtr.*sqrt(Pr_mr.^3./(pt.*x3.^3));
% Total P_req including accessories and losses
Pr = (Pr_mr/etaMR+Pr_tr/etaTR+HPacc*550)/etaCBOX;
pr1 = Pr./Poei;

%--- Calculate Governed Engine Power Available --%
p2g = min(pr1 - G1.*(x3-1),1);
p2g(p2g<0) = 0;

%---- Calculate Derivatives ----------------------%
dx1p = g0 - m0.*x3.^2.*x8.*(kg-fv*fw).*cos(x9) - ...
       f0z.*x1.*sqrt(x1.^2+x2.^2);
dx2p =      m0.*x3.^2.*x8.*(kg-fv*fw).*sin(x9) - ...
       f0x.*x2.*sqrt(x1.^2+x2.^2);
dx3p = i0./x3.*(x6+x7-pr1);
dx4p = -0.1.*x1;
dx5p = 0.1.*x2;
dx6p = -k1.*x6;
dx7p = zeros(length(x1),1);
for nn = 1:length(x1)
    if t(nn) > .1548
    dx7p(nn) = k2.*(p2g(nn)-x7(nn));
    end
end
dx8p  = tref.*x10;
dx9p  = tref.*x11;
dx10p = tref.*u1;
dx11p = tref.*u2;
dxp = [dx1p, dx2p, dx3p, dx4p, dx5p, ...
       dx6p, dx7p, dx8p, dx9p, dx10p, dx11p]';

%-------------------------------------------------%
% -------- End Function:  AH-1Z Dynamics -------- %
%-------------------------------------------------%
```

D.4 Induced Velocity Functions

D.4.1 Baseline Induced Velocity

```
function [lambda_i, dlambda_i] = getInducedVelocityBaseline3(mu_x, mu_z, mu_zAID, mu_zBID, VRS)
%
% Calculate induced velocity and induced velocity slope from the baseline
% curve as defined by Johnson in NASA/TP-2005-213477.
%
% Input:
%   mu_x    = nondimensional rotor horizontal velocity, V_x/v_h
%   mu_z    = nondimensional rotor vertical velocity, by V_z/v_h
%   mu_zAID = nondimensional rotor vertical velocity at point A, V_zAID/v_h
%   mu_zBID = nondimensional rotor vertical velocity at point B, V_zBID/v_h
%   VRS     = vortex ring state parameters
```

```
% Output:
%   lambda_i  = nondimensional rotor induced velocity, v_i/v_h
%   dlambda_i = nondimensional slope of rotor induced velocity,
%               dlambda_i/dmu_z

  if(mu_z >= 0.0 || mu_x >= VRS.mu_xC)
    [lambda_i, dlambda_i] = getInducedVelocityMomentum3(mu_x, mu_z);
  else
    if(mu_zBID < mu_z && mu_z < mu_zAID)
      [lambdaA, dlambdaA] = getInducedVelocityMomentum3(mu_x, mu_zAID);
      [lambdaB, ~] = getInducedVelocityMomentum3(mu_x, mu_zBID);
      coeffB = solveInducedVelocity(3, mu_zAID, mu_zBID, lambdaA, dlambdaA, lambdaB, 0.0);
      b = coeffB(2);
      c = coeffB(3);
      d = coeffB(4);
      lambda_i = mu_z*(b + mu_z*(c + mu_z*d));
      dlambda_i = b + mu_z*(2.0*c + 3.0*mu_z*d);
    else
      [lambda_i, dlambda_i] = getInducedVelocityMomentum3(mu_x, mu_z);
    end
  end
end
```

D.4.2 Momentum Theory Induced Velocity

```
function [lambda_i, dlambda_i] = getInducedVelocityMomentum3(mu_x, mu_z)
%
% Calculate induced velocity and induced velocity slope using momentum theory
%
% Input:
%   mu_x = nondimensional rotor horizontal velocity, V_x/v_h
%   mu_z = nondimensional rotor vertical velocity, V_z/v_h
% Output:
%   lambda_i  = nondimensional rotor induced velocity, v_i/v_h
%   dlambda_i = nondimensional slope of rotor induced velocity,
%               dlambda_i)/dmu_z
%
  if(mu_x < 0.000001)
    if(mu_z > -2.0)
      lambda_i = -(0.5*mu_z) + sqrt((0.5*mu_z)^2 + 1.0);
      dlambda_i = -0.5 + 0.25*mu_z/sqrt((0.5*mu_z)^2 + 1.0);
    elseif(mu_z == -2.0)
      lambda_i = 1.0;
      dlambda_i = 10000.0;
    else
      lambda_i = -(0.5*mu_z) - sqrt((0.5*mu_z)^2 - 1.0);
      dlambda_i = -0.5 - 0.25*mu_z/sqrt((0.5*mu_z)^2 - 1.0);
    end
  else
    lambda = newtonraphsonlambda(mu_x,mu_z);
    lambda_i = lambda-mu_z;
%     fInduced = @(lambda_i)(1 - lambda_i*sqrt(mu_x^2 + (mu_z + lambda_i)^2));
%     lambda_i = fzero(fInduced, 1);
    dlambda_i = -lambda_i*(mu_z + lambda_i)/(mu_x^2 + (mu_z + lambda_i)*(mu_z + 2.0*lambda_i));
  end
end
```

D.4.2.1 Newton-Raphson Solution of the Momentum Quartic

```
function lam = newtonraphsonlambda(mux, muz)
```

```
% Function to calculate induced velocity
% Follows Johnson's Textbook page 126

% mu_x = ucos(alpha)+wsin(alpha)/vh
% mu_z = usin(alpha)-wcos(alpha)/vh

f = 0.5;    % Relaxation factor for improved convergence
tol = 1e-6; % convergence tolerance
err = 1;
lam = (mux^2+(1+muz)^2)^-.5+muz;

while err >= tol

lam_i = (mux^2+lam^2)^-.5;
lam_new = lam - (lam - muz - lam_i)/(1+lam_i*lam/(lam^2+mux^2))*f;
err = abs(lam_new-lam);
lam = lam_new;

end
```

D.4.3 VRS Region Induced Velocity

```
function lambda_i = getInducedVelocityVRS3(mu_x, mu_z,VRS_On)
%
%  Calculate the rotor induced velocity using Johnson's method from
%  NASA/TP-2005-213477.
%
%  Input:
%    mu_x = nondimensional rotor horizontal velocity, V_x/v_h
%    mu_z = nondimensional rotor vertical velocity, V_z/v_h
%    VRS_On = VRS flag (True = VRS on, False = VRS off)
%  Output:
%    lambda_i = nondimensional rotor induced velocity, v_i/v_h
%
  % VRS constants
  VRS.mu_zA = -1.5;
  VRS.mu_zB = -2.1;
  VRS.mu_xC = 0.75;
  VRS.mu_zD = -0.2;
  VRS.mu_zN = -0.45;
  VRS.lambda_N = 0.85;
  VRS.mu_zX = -1.5;
  VRS.lambda_X = 1.25;
  VRS.mu_zE = -2.0;
  VRS.mu_xM = 0.95;
  VRS.f = 1.0;
%
% Momentum theory and baseline curve
  mu_zAID = VRS.mu_zA + 0.2*(mu_x/VRS.mu_xC)^2;
  mu_zBID = VRS.mu_zB + 0.2*(mu_x/VRS.mu_xC)^2;
  if(mu_x/VRS.mu_xC > 0.5)
    mu_zBID = mu_zBID + 0.7*(mu_zAID - mu_zBID)*(2.0*mu_x/VRS.mu_xC - 1.0)^3;
  end
  [lambda_i, ~] = getInducedVelocityBaseline3(mu_x, mu_z, mu_zAID, mu_zBID,VRS);
%
% Vortex ring state model
  if(VRS_On && mu_z < 0.0 && mu_x < VRS.mu_xM)
    mu_zDID = VRS.mu_zD;
    mu_zNID = 0.5*(VRS.mu_zN + VRS.mu_zX) + 0.5*(VRS.mu_zN - VRS.mu_zX)*(1.0 - (mu_x/VRS.mu_xM)^2)^0.2;
    mu_zXID = 0.5*(VRS.mu_zN + VRS.mu_zX) - 0.5*(VRS.mu_zN - VRS.mu_zX)*(1.0 - (mu_x/VRS.mu_xM)^2)^1.5;
    mu_zEID = VRS.mu_zE + (mu_zXID - VRS.mu_zX);
%
```

```
    if(mu_zEID < mu_z && mu_z < mu_zDID)
      if(mu_zNID <= mu_z && mu_z < mu_zDID)
        DlambdaD = 0.0;
        DdlambdaD = 0.0;
        [lambdaNmom, ~] = getInducedVelocityMomentum3(0.0, VRS.mu_zN);
        DlambdaN = (VRS.lambda_N - (VRS.mu_zN + lambdaNmom))*sqrt(1.0 - (mu_x/VRS.mu_xM)^6);
        [~, dlambdaNID] = getInducedVelocityBaseline3(mu_x, mu_zNID, mu_zAID, mu_zBID, VRS);
        DdlambdaN = -(1.0 + dlambdaNID);
        coeffV = solveInducedVelocity(4, mu_zDID, mu_zNID, DlambdaD, DdlambdaD, DlambdaN, DdlambdaN);
      elseif(mu_zXID <= mu_z && mu_z < mu_zNID)
        [lambdaNmom, ~] = getInducedVelocityMomentum3(0.0, VRS.mu_zN);
        DlambdaN = (VRS.lambda_N - (VRS.mu_zN + lambdaNmom))*sqrt(1.0 - (mu_x/VRS.mu_xM)^6);
        [lambdaXmom, ~] = getInducedVelocityMomentum3(0.0, VRS.mu_zX);
        DlambdaX = (VRS.lambda_X - (VRS.mu_zX + lambdaXmom))*sqrt(1.0 - (mu_x/VRS.mu_xM)^6);
        [~, dlambdaNID] = getInducedVelocityBaseline3(mu_x, mu_zNID, mu_zAID, mu_zBID, VRS);
        DdlambdaN = -(1.0 + dlambdaNID);
        [~, dlambdaXID] = getInducedVelocityBaseline3(mu_x, mu_zXID, mu_zAID, mu_zBID, VRS);
        DdlambdaX = -(1.0 + dlambdaXID);
        coeffV = solveInducedVelocity(4, mu_zNID, mu_zXID, DlambdaN, DdlambdaN, DlambdaX, DdlambdaX);
      else
        [lambdaXmom, ~] = getInducedVelocityMomentum3(0.0, VRS.mu_zX);
        DlambdaX = (VRS.lambda_X - (VRS.mu_zX + lambdaXmom))*sqrt(1.0 - (mu_x/VRS.mu_xM)^6);
        [~, dlambdaXID] = getInducedVelocityBaseline3(mu_x, mu_zXID, mu_zAID, mu_zBID, VRS);
        DdlambdaX = -(1.0 + dlambdaXID);
        DlambdaE = 0.0;
        coeffV = solveInducedVelocity(3, mu_zXID, mu_zEID, DlambdaX, DdlambdaX, DlambdaE, 0.0);
      end
      a = coeffV(1);
      b = coeffV(2);
      c = coeffV(3);
      d = coeffV(4);
      DeltaVRS = a + mu_z*(b + mu_z*(c + mu_z*d));
      lambda_i = lambda_i + VRS.f*DeltaVRS;
    end
  end
end
```

D.4.4 Solve the Induced Velocity

```
function coeff = solveInducedVelocity(n, mu_zA, mu_zB, lambdaA, dlambdaA, lambdaB, dlambdaB)
%
  coeff = zeros(1, 4);
  mtxA = [[1.0, mu_zA, mu_zA^2,   mu_zA^3   ];
          [0.0, 1.0,   2.0*mu_zA, 3.0*mu_zA^2];
          [1.0, mu_zB, mu_zB^2,   mu_zB^3   ];
          [0.0, 1.0,   2.0*mu_zB, 3.0*mu_zB^2]];
  vecB = [lambdaA ;
          dlambdaA;
          lambdaB ;
          dlambdaB];
  if(n == 3)
    coeff(2:4) = mtxA(1:3,2:4)\vecB(1:3);
  else
    coeff = mtxA\vecB;
  end
end
```

D.5 GPOPS-II® Main File and Functions

D.5.1 GPOPS-II® Main File

```
%-------------------------------------------------------------------------%
%---------------------- Two Phase Problem ------------------------%
%-------------------------------------------------------------------------%
clear all; clc;

%-------------------------------------------------------------------------%
%------------------------ Aircraft Inputs --------------------------%
%-------------------------------------------------------------------------%
weight  = 16200;        % lbf   17940 gives power ratio of 0.7
omega0  = 30.0546;      % Nominal Main Rotor speed (rad/sec)
                        % (for early prototypes, 287 rpm)
R       = 24;           % Main Rotor Radius (ft)
Rt      = 4.875;        % Tail Rotor Radius (ft)
Rcut    = 6;            % Rotor cutout (cuff radius)
tilt    = 3;            % Mast tilt in degrees
fex     = 24.7;         % ft^2  from design report
fez     = 168.14;       % Total equivalent planform flat plate area
fezv    = 86.44;        % Equivalent flat plate under rotor disk
Ir      = 1362.3*4;     % slug-ft^2 from
sigma   = 0.1026;       % Solidity Ratio
cd      = 0.0078;       % Average blade coefficient of drag
ki      = 1.08;         % Induced velocity correction factor
M       = .88;          % Main rotor efficiency
Mtr     = .70;          % Tail rotor efficienty
etaMR   = .984;         % Main rotor drive efficiency
etaTR   = .98;          % Tail rotor drive efficiency
etaCBOX = .986;         % Combining gearbox efficiency
HPacc   = 25;           % Accessory drive horsepower
loss    = 1.08;         % Pr increase due to drivetrain/accessories
ltr     = 29.343;       % TR moment arm
Poei    = 1580*550;     % ft-lb/s (from HV Report)
G       = 6e4;          % DECU gain
tau1    = .5;           % failed engine time constant
tau2    = .4;           % OEI engine time constant
hh      = 12.33;        % Rotor hub height (for ground effect calc)
deltafe = 10;           % fe_x fudge factor to make cp curves match (USED 7 EARLIER)

% ---------------------- Derived Constants ------------------------ %
g       = 32.174;                           % gravitational accel
rho     = 0.002377;                         % air density
mass    = weight/g;                         % mass in slugs
tref    = 100/omega0;                       % non-dim ref time
g0      = 1e4*g/(omega0^2*R);               % for x1 equation
m0      = 10*rho*pi*R^3/mass;               % for x2 equation
f0z     = rho*fez*R/(2*mass);               % for x1 equation
f0x     = rho*fex*R/(2*mass);               % for x2 equation
i0      = 100*Poei/(Ir*omega0^3);           % for x3 equation
k1      = 100/(omega0*tau1);                % for x6 equation
k2      = 100/(omega0*tau2);                % for x7 equation
G1      = G*omega0/Poei;                    % equiv. gain for eng2 pwr
p1      = sigma*cd/8;                       % Cp term
p2      = (fex+deltafe)/(2*pi*R^2);         % Cp term
p3      = .01*weight/(rho*pi*R^4*omega0^2); % Cp term
p4      = rho*pi*R^5*omega0^3;              % Cp to Power term
pt      = 2*rho*pi*Rt^2*ltr^3*omega0^3;     % Cp_t term
fv      = fezv/(pi*R^2-pi*Rcut^2);          % Vertical drag factor
mtilt   = deg2rad(tilt);                    % Mast tilt in radians

%-----------------User Input for new VRS Curve Fits-----------------%
useVRS  = true;
```

```
%------------------------ Washout Airspeed -------------------------%
% The speed at which rotor wash no longer impenges on the fuselage
v1 = 30;           % knots
x2w = 1.68781*v1/(.01*omega0*R);

%----------------- Data Needed for Endpoint Fx J2 ------------------%
href = 30;                          % Obj function reference height
x4r = href/(10*R);

% Altitude where phases changeover
x4chng = 3/(10*R);

%-------------------------------------------------------------------%
%-------------------- Store constants in AUXDATA -------------------%
%-------------------------------------------------------------------%
auxdata.useVRS  = useVRS;
auxdata.g0      = g0;
auxdata.m0      = m0;
auxdata.f0z     = f0z;
auxdata.f0x     = f0x;
auxdata.i0      = i0;
auxdata.k1      = k1;
auxdata.k2      = k2;
auxdata.G1      = G1;
auxdata.hh      = hh;
auxdata.omega0  = omega0;
auxdata.R       = R;
auxdata.M       = M;
auxdata.Mtr     = Mtr;
auxdata.etaMR   = etaMR;
auxdata.etaTR   = etaTR;
auxdata.etaCBOX = etaCBOX;
auxdata.HPacc   = HPacc;
auxdata.mtilt   = mtilt;
auxdata.ki      = ki;
auxdata.p1      = p1;
auxdata.p2      = p2;
auxdata.p3      = p3;
auxdata.p4      = p4;
auxdata.pt      = pt;
auxdata.Poei    = Poei;
auxdata.loss    = loss;
auxdata.fv      = fv;
auxdata.x2w     = x2w;
auxdata.x4r     = x4r;
auxdata.tref    = tref;
auxdata.x4chng  = x4chng;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%---------------------------- KNOBS ------------------------------%
%-------------------------------------------------------------------%
%                                                                   %
%                                                                   %
    h0guess     = 136;          % Altutude guess must be close %30
                                % for accurate ground effect.
                                % Iterate if required.

    start_aspd  = 0;

    tauf_g      = 15;           % Change guess time %2
    maxx1g      =  8;
    maxx2g      = 10;           % Guess for max fwd airspeed %2
    meshtol     = 1e-4;         % Mesh Tolerance
    nlptol      = 1e-6;         % SNOPT tolerance
    deriv_size  = 1e-8;         % Derivative Step Size
    h0_low      =  30;          % Lowest available altitude
```

```
    h0_high     = 300;          % Highest available altitude

    % Objective function weight
%     Wt          = h0guess/(100*R*tauf_g);
    Wt          = 0;
    WL          = .001; %0.01;
%     WL          = 0;
    W1          = .01; %.002;
    W2          = .5; %.0125;
    auxdata.Wt  = Wt;
    auxdata.WL  = WL;
    auxdata.W1  = W1;
    auxdata.W2  = W2;

% Max Ct
Ctmax = .015;
% Max rate of change for Ct
dctmax = 1;
dctmin = -1;
% Max TPP angle in deg
TPPlo = deg2rad(-11);
TPPhi = deg2rad(11);
% Max rate of change of TPP angle in deg/sec
dTPPlo = deg2rad(-12);
dTPPhi = deg2rad(12);
%---- Path Bounds (Controls)
% Max accel of Ct*1000
d2ctmin = -21;
d2ctmax = 21; %5.75
% Max accel of TPP angle
d2TPPlo = -deg2rad(15);  % 10  15
d2TPPhi = deg2rad(15); % 14.4  15

%                                                                              %
%                                                                              %
%------------------------------------------------------------------------------%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%------------------------------------------------------------------------------%
%------------------------ Initial Conditions ----------------------%
%------------------------------------------------------------------------------%

% Given initial conditions (Steady Level Flight for this case)
x1_0    = 0;
x3_0    = 1;
x4_0    = h0guess/(10*R);
x5_0    = 0;
u1_0    = 0;
u2_0    = 0;

% Guess remainder of initial conditions
x6_0g   = 0.5;
x7_0g   = x6_0g;
x8_0g   = 6;
x9_0g   = 0;

%------------------%
% Initial airspeed %
%------------------%
% Setmanually or by a loop:
x2N = start_aspd*1.68781./(.01*omega0*R);

n = 1;
x2_0 = x2N(n);
%------------------%

%---- Calculate initial engine power states -----------------------%
```

```
y0 = [x7_0g x8_0g x9_0g];
x0 = [x1_0 x2_0 x3_0 x4_0 x5_0];

ic = fsolve(@(x) AH1init_v2o(x,x0,auxdata),y0)
x6_0 = ic(1); x7_0 = x6_0;
x8_0 = ic(2);
x9_0 = ic(3);
x10_0 = 0;
x11_0 = 0;
%%
%---- Pilot Delay ------------------------------------------------%
% Set Pilot delay time:
tdelay = 1.5;
taudelay = tdelay/tref;
auxdata.taudelay = taudelay;
%---- Provide values for d/dt(Ct) and d/dt(alpha) during delay -----%
u1del = 0;
u2del = 0;
% Initial conditions for ode45 integration:
x0 = [x1_0,x2_0,x3_0,x4_0,x5_0,x6_0,x7_0,x8_0,x9_0,x10_0,x11_0];
tauspan = [0 taudelay];

options = odeset('AbsTol',1e-5);
[taui,Y] = ode45(@(tau,Y) ah1zdelay13(tau,Y,u1del,u2del,auxdata),tauspan,x0,options);

%           Plot pilot delay integration if needed:
%           plot(taui,Y(:,1),taui,Y(:,2),taui,Y(:,3),...
%           taui,Y(:,4),taui,Y(:,5),taui,Y(:,6),taui,Y(:,7))

%---- Subscript "1" denotes initial conditions post ode45 delay ----%
x1_1 = Y(end,1);
x2_1 = Y(end,2);
x3_1 = Y(end,3);
x4_1 = Y(end,4);
x5_1 = Y(end,5);
x6_1 = Y(end,6);
x7_1 = Y(end,7);
x8_1 = Y(end,8);
x9_1 = Y(end,9);
x10_1 = Y(end,10);
x11_1 = Y(end,11);
x40delta = x4_0-x4_1;       % how far the aircraft falls in the delay

%---------------------- End Initial Conditions ---------------------%

%%

%-------------------------------------------------------------------%
%----------------------- Limits on Variables -----------------------%
%-------------------------------------------------------------------%

%---- Bounds on initial state ------------------------------------%
% CHANGE h0_low depending on whether you are finding
% the upper or lower half of HV curve
x4_0lo      = h0_low/(10*R);
x4_0hi      = h0_high/(10*R);

%---- Non-Dimensional Time Limits --------------------------------%
tau0    = 0;
% tau0    = taudelay;
taufmin = 1;        taufmax     = 20;

%---- Path Bounds (States) ---------------------------------------%
% Max Rate of Descent in ft/min
maxROD  = 3000;
% Max airspeed (groundspeed) in knots
vmax = 150;
% Max and min rotor RPM (in fraction)
RPMmin = .92; RPMmax = 1.06;
```

```
% Max horizontal distance (ft)
xmax = 10000;


%---- Bounds in Phase 1 ------------------------------------------%
x1lo    = 0;        x1hi        = maxROD/60/(.01*R*omega0);
x2lo    = 0;        x2hi        = vmax*1.68781/(.01*omega0*R);
x3lo    = RPMmin;   x3hi        = RPMmax;
x4lo    = 0;        x4hi        = x4_0hi;
x5lo    = 0;        x5hi        = xmax/(10*R);
x6lo    = 0;        x6hi        = 1;    % Engine power 0 to 100%
x7lo    = 0;        x7hi        = 1;    % Engine power 0 to 100%
x8lo    = 0;        x8hi        = Ctmax*1000;
x9lo    = TPPlo;    x9hi        = TPPhi;
x10lo   = dctmin;   x10hi       = dctmax;
x11lo   = dTPPlo;   x11hi       = dTPPhi;

u1lo    = d2ctmin;
u1hi    = d2ctmax;
u2lo    = d2TPPlo;
u2hi    = d2TPPhi;

%---- Bounds in Phase 2 ------------------------------------------%
% Max TPP angle
TPP1p2 = 5;
TPPp2 = deg2rad(TPP1p2);
x9lop2 = -TPPp2;
x9hip2 = TPPp2;

%---- Bounds on final state --------------------------------------%
% Max vertical Velocity (ft/sec)
maxfinalsink = 1;
% Max final groundspeed (knots)
maxfinalgrnd = 10;
x1fhi   = maxfinalsink/(.01*R*omega0);
x2fhi   = maxfinalgrnd*1.68781/(.01*R*omega0);
TPPflo = -3;
TPPfhi = 0;
x9flo = deg2rad(TPPflo);
x9fhi = deg2rad(TPPfhi);
%------------------------------------------------------------------%
%---------------------- Phase 1 Bounds and Guess ------------------%
%------------------------------------------------------------------%
bounds.phase(1).initialtime.lower  = tau0;
bounds.phase(1).initialtime.upper  = tau0;

bounds.phase(1).finaltime.lower    = tau0;
bounds.phase(1).finaltime.upper    = taufmax;

bounds.phase(1).initialstate.lower = ...
[x1_1, x2_1, x3_1, x4_0lo, x5_1, x6_1, x7_1, x8_1, x9_1, x10_1, x11_1];
bounds.phase(1).initialstate.upper = ...
[x1_1, x2_1, x3_1, x4_0hi, x5_1, x6_1, x7_1, x8_1, x9_1, x10_1, x11_1];

bounds.phase(1).state.lower = ...
[x1lo, x2lo, x3lo, x4lo, x5lo, x6lo, x7lo, x8lo, x9lo, x10lo, x11lo];
bounds.phase(1).state.upper = ...
[x1hi, x2hi, x3hi, x4hi, x5hi, x6hi, x7hi, x8hi, x9hi, x10hi, x11hi];

bounds.phase(1).finalstate.lower   = ...
[x1lo, x2lo, x3lo, x4chng, x5lo, x6lo, x7lo, x8lo, x9lop2, x10lo, x11lo];
bounds.phase(1).finalstate.upper   = ...
[x1hi, x2hi, x3hi, x4chng, x5hi, x6hi, x7hi, x8hi, x9hip2, x10hi, x11hi];

bounds.phase(1).control.lower      = [u1lo, u2lo];
bounds.phase(1).control.upper      = [u1hi, u2hi];

bounds.phase(1).integral.lower     = 0;
bounds.phase(1).integral.upper     = 1000;
```

```
guess.phase(1).time               = [0; tauf_g/2;  tauf_g];
guess.phase(1).state(:,1)         = [x1_1; maxx1g; 0];
guess.phase(1).state(:,2)         = [x2_1; maxx2g; 2]; %[x2_1; maxx2g; 2];
guess.phase(1).state(:,3)         = [x3_1; 0.98; 0.95];
guess.phase(1).state(:,4)         = [x4_0; x4_1/2; x4_1/4];
guess.phase(1).state(:,5)         = [x5_1; 1; 3];
guess.phase(1).state(:,6)         = [x6_1; 0; 0];
guess.phase(1).state(:,7)         = [x7_1; 1; 1];
guess.phase(1).state(:,8)         = [x8_1; x8_1; x8_1];
guess.phase(1).state(:,9)         = [x9_1; x9_1; x9_1];
guess.phase(1).state(:,10)        = [x10_1; x10_1; x10_1];
guess.phase(1).state(:,11)        = [x11_1; x11_1; x11_1];
guess.phase(1).control(:,1)       = [1; 1; 1];
guess.phase(1).control(:,2)       = [0; 0; 0];

guess.phase(1).integral           = 1;

%------------------------------------------------------------------%
%--------------------- Phase 2 Bounds and Guess -------------------%
%------------------------------------------------------------------%
bounds.phase(2).initialtime.lower  = taufmin;
bounds.phase(2).initialtime.upper  = taufmax;

bounds.phase(2).finaltime.lower    = taufmin;
bounds.phase(2).finaltime.upper    = taufmax;

bounds.phase(2).initialstate.lower = ...
[x1lo, x2lo, x3lo, x4chng, x5lo, x6lo, x7lo, x8lo, x9lop2, x10lo, x11lo];
bounds.phase(2).initialstate.upper = ...
[x1hi, x2hi, x3hi, x4chng, x5hi, x6hi, x7hi, x8hi, x9hip2, x10hi, x11hi];

bounds.phase(2).state.lower = ...
[x1lo, x2lo, x3lo, x4lo, x5lo, x6lo, x7lo, x8lo, x9lop2, x10lo, x11lo];
bounds.phase(2).state.upper = ...
[x1hi, x2hi, x3hi, x4hi, x5hi, x6hi, x7hi, x8hi, x9hip2, x10hi, x11hi];

bounds.phase(2).finalstate.lower   = ...
[0,     0,     x3lo, 0, x5lo, x6lo, x7lo, x8lo, x9flo, x10lo, x11lo];
bounds.phase(2).finalstate.upper   = ...
[x1fhi, x2fhi, x3hi, 0, x5hi, x6hi, x7hi, x8hi, x9fhi, x10hi, x11hi];

bounds.phase(2).control.lower      = [u1lo, u2lo];
bounds.phase(2).control.upper      = [u1hi, u2hi];

bounds.phase(2).integral.lower     = 0;
bounds.phase(2).integral.upper     = 10000;

guess.phase(2).time               = [tauf_g+.1; tauf_g+.5; tauf_g+1];
guess.phase(2).state(:,1)         = [0; 1; 0];
guess.phase(2).state(:,2)         = [2; 1; 0]; % [2.1; 1; 0];
guess.phase(2).state(:,3)         = [.96; .95; .92];
guess.phase(2).state(:,4)         = [x4_1/5; .05; 0];
guess.phase(2).state(:,5)         = [3.1; 3.5; 4];
guess.phase(2).state(:,6)         = [0; 0; 0];
guess.phase(2).state(:,7)         = [.9; 1; 1];
guess.phase(2).state(:,8)         = [x8_1; x8_1; x8_1];
guess.phase(2).state(:,9)         = [x9lop2; x9lop2; x9fhi];
guess.phase(2).state(:,10)        = [x10_1; x10_1; x10_1];
guess.phase(2).state(:,11)        = [x11_1; x11_1; x11_1];
guess.phase(2).control(:,1)       = [1; 1; 1];
guess.phase(2).control(:,2)       = [0; 0; 0];

guess.phase(2).integral           = 1;

%------------------------------------------------------------------------%
%------------- Set up Event Constraints That Link Phases ----------------%
%------------------------------------------------------------------------%
bounds.eventgroup(1).lower = zeros(1,12);
```

```
bounds.eventgroup(1).upper = zeros(1,12);

% bounds.eventgroup(2).lower = 0;
% bounds.eventgroup(2).upper = 0;

bounds.eventgroup(2).lower = 0.01*ones(1,2);
bounds.eventgroup(2).upper = 12*ones(1,2);


%---------------------------------------------------------------------------%
%-------------------- Provide Mesh Refinement Method -----------------------%
%---------------------------------------------------------------------------%
mesh.method       = 'hp-LiuRao-Legendre';
% mesh.method       = 'hp-DarbyRao';
% mesh.method       = 'hp-PattersonRao';
% mesh.method       = 'hp-LiuRao';
mesh.tolerance    = meshtol;
mesh.colpointsmin = 4;
mesh.colpointsmax = 30;
mesh.sigma        = 0.5;

%---------------------------------------------------------------------%
%---------- Configure Setup Using the information provided ---------%
%---------------------------------------------------------------------%
setup.name                           = 'GPOPSAH1Z5';
setup.functions.continuous           = @Cont13;
setup.functions.endpoint             = @J13;
setup.displaylevel                   = 2;
setup.nlp.solver                     = 'snopt';
setup.nlp.snoptoptions.tolerance     = nlptol; % 1e-09
setup.nlp.snoptoptions.maxiterations = 2000;
setup.mesh                           = mesh;
setup.bounds                         = bounds;
setup.guess                          = guess;
setup.auxdata                        = auxdata;
setup.derivatives.stepsize1          = deriv_size;
setup.derivatives.supplier           = 'sparseFD';
setup.derivatives.derivativelevel    = 'first';
setup.derivatives.dependencies       = 'full';
setup.scales.method                  = 'none'; %'automatic-guessUpdate';
setup.method                         = 'RPM-Differentiation';
setup.mesh.maxiterations             = 20;

%---------------------------------------------------------------------%
%------------------- Solve Problem Using GPOPS2 ----------------------%
%---------------------------------------------------------------------%
output = gpops2(setup);

disp(['Finished with u_0 = ',num2str(x2N(n)/1.68781*.01*omega0*R)]);


%%
% Extract Solution

Xp1 = output.result.solution.phase(1).state;
Xp2 = output.result.solution.phase(2).state;
X = [Xp1(1:length(Xp1)-1,:); Xp2];

Up1 = output.result.solution.phase(1).control;
u1p1 = Up1(:,1); u2p1 = Up1(:,2);
Up2 = output.result.solution.phase(2).control;
u1p2 = Up2(:,1); u2p2 = Up2(:,2);
U = [Up1(1:length(Up1)-1,:); Up2];
u1 = U(:,1); u2 = U(:,2);

taup1 = output.result.solution.phase(1).time + taudelay;
taup2 = output.result.solution.phase(2).time + taudelay;
tau = [taup1(1:length(taup1)-1);taup2];
```

```
costatesp1 = output.result.solution.phase(1).costate;
costatesp2 = output.result.solution.phase(2).costate;
costates = [costatesp1(1:length(costatesp1)-1,:);costatesp2];

% Print the solution set thus far
    u0ans(n) =  output.result.solution.phase(1).state(1,2)*.01*omega0*R/1.68781
    h0ans(n) = (output.result.solution.phase(1).state(1,4)+x40delta)*10*R
%%
% Save the hamiltonian for analysis
H1 = zeros(size(costatesp1,1),1);
H2 = zeros(size(costatesp2,1),1);
nz1 = zeros(size(taup1,1),1);
nz2 = zeros(size(taup2,1),1);

for j = 1:size(costatesp1,1)

    t = taup1(j);
    Xx = Xp1(j,:);

    XDOT = ah1zdelay13(t,Xx,u1p1(j),u2p1(j),auxdata);
    H1(j) = costatesp1(j,:)*XDOT;
    nz1(j) = cos(Xx(9)) - 1/g0.*(XDOT(2).*sin(Xx(9))-XDOT(1).*cos(Xx(9)));
end

for j = 1:size(costatesp2,1)

    t = taup2(j);
    Xx = Xp2(j,:);

    XDOT = ah1zdelay13(t,Xx,u1p2(j),u2p2(j),auxdata);
    H2(j) = costatesp2(j,:)*XDOT;
    nz2(j) = cos(Xx(9)) - 1/g0.*(XDOT(2).*sin(Xx(9))-XDOT(1).*cos(Xx(9)));
end
nz = [nz1(1:length(nz1)-1);nz2];

H = [H1(1:length(H1)-1,:);H2];

maxH = max([max(abs(H1)), max(abs(H2))]);
tau_phxc = taup1(end)+taudelay;
% figure;
% plot(taup1,H1,'o',taup2,H2,'o'); grid on;
% figure;
% plot(tau,nz)
```

D.5.2 GPOPS-II® Continuous File

```
%-----------------------------------------------%
% Begin Function:  GPOPSAH1ZCont10.m  %
%-----------------------------------------------%
function output = Cont13(input)

const   = input.auxdata;
useVRS  = const.useVRS;
g0      = const.g0;
m0      = const.m0;
f0z     = const.f0z;
f0x     = const.f0x;
i0      = const.i0;
k1      = const.k1;
k2      = const.k2;
G1      = const.G1;
M       = const.M;
```

```
Mtr     = const.Mtr;
etaMR   = const.etaMR;
etaTR   = const.etaTR;
etaCBOX = const.etaCBOX;
HPacc   = const.HPacc;
ki      = const.ki;
p1      = const.p1;
p2      = const.p2;
p3      = const.p3;
p4      = const.p4;
pt      = const.pt;
Poei    = const.Poei;
R       = const.R;
hh      = const.hh;
fv      = const.fv;
x2w     = const.x2w;
taudelay = const.taudelay;
W1      = const.W1;
W2      = const.W2;
tref    = const.tref;

%-------------------------------------------------%
%--- Phase 1 -------------------------------------%
%-------------------------------------------------%
x1      = input.phase(1).state(:,1);
x2      = input.phase(1).state(:,2);
x3      = input.phase(1).state(:,3);
x4      = input.phase(1).state(:,4);

x6      = input.phase(1).state(:,6);
x7      = input.phase(1).state(:,7);
x8      = input.phase(1).state(:,8);
x9      = input.phase(1).state(:,9);
x10      = input.phase(1).state(:,10);
x11      = input.phase(1).state(:,11);

u1      = input.phase(1).control(:,1);
u2      = input.phase(1).control(:,2);

taup1     = input.phase(1).time + taudelay;


%---- Calculate Ground Effect --------------------%
z = max(hh+10.*R.*x4,hh);
% Hayden
kg = (0.9926 + 0.03794.*(z./(2*R)).^-2).^(2/3);
kg(z>(2*R)) = 1;

%---- Calculate Washout --------------------------%
fw = 1-1/x2w.*abs(x2);
fw(x2>x2w) = 0;

%---- Aerodynamic Model---------------------------%
% Calculate Johnson's mux = Vx_rotor/vh, muz = -Vz_rotor/vh
% updated with revised equations for mu
mux = 0.4472./(x3.*sqrt(x8)).*(x2.*cos(x9) + x1.*sin(x9));
muz = 0.4472./(x3.*sqrt(x8)).*(x2.*sin(x9) - x1.*cos(x9));

lam_i_J = ones(length(x1),1);

for m = 1:length(x1)
% Call getInducedVelocityVRS.m to obtain lambda_i
lam_i_J(m) = getInducedVelocityVRS3(mux(m),muz(m),useVRS);
end

% lam_i = zeros(length(x1),1);
% for mm = 1:length(x1)
%     if (2*muz(mm)+3)^2+mux(mm)^2 > 1.0
%         lambda = newtonraphsonlambda(mux(mm),muz(mm));
```

```
%         lam_iJ = lambda-muz(mm);
%         lam_i(mm) = 0.0224.*lam_iJ.*(x8(mm)).^.5;
%     else
%         lam_i(mm) = muz(mm)*(0.373*muz(mm)^2+0.598*mux(mm)^2-1.991);
%     end
% end


% Regular mu, and lambda_i
mu  = 0.01./x3.*(x2.*cos(x9) + x1.*sin(x9));
lam_i = 0.0224.*lam_i_J.*(x8).^.5;

%---- End Aerodynamic Model----------------------%


%---- Calculate Power Requred -------------------%
% Main Rotor
Cp_mr = ki.*lam_i.*x8./1000 + p1.*(1+4.65.*mu.^2) + p2.*mu.^3 - p3.*x1./x3.^3;
Pr_mr = 1/M.*p4.*Cp_mr.*x3.^3;
% Tail Rotor
Pr_tr = (1-1.2*mu).*1./Mtr.*sqrt(Pr_mr.^3./(pt.*x3.^3));


% Total P_req including accessories and losses
Pr = (Pr_mr./etaMR+Pr_tr./etaTR+HPacc.*550)./etaCBOX;
pr1 = Pr./Poei;


%--- Calculate Governed Engine Power Available --%
p2g = min(pr1 - G1.*(x3-1),1);
p2g(p2g<0) = 0;

%---- Calculate Derivatives ----------------------%
dx1p = g0 - m0.*x3.^2.*x8.*(kg-fv.*fw).*cos(x9) - ...
       f0z.*x1.*sqrt(x1.^2+x2.^2);
dx2p =      m0.*x3.^2.*x8.*(kg-fv.*fw).*sin(x9) - ...
       f0x.*x2.*sqrt(x1.^2+x2.^2);
dx3p = i0./x3.*(x6+x7-pr1);
dx4p = -0.1.*x1;
dx5p = 0.1.*x2;
dx6p = -k1.*x6;
dx7p = zeros(length(x1),1);
for nn = 1:length(x1)
    if taup1(nn) > .1548
    dx7p(nn) = k2.*(p2g(nn)-x7(nn));
    end
end
dx8p  = tref.*x10;
dx9p  = tref.*x11;
dx10p = tref.*u1;
dx11p = tref.*u2;
output(1).dynamics = [dx1p, dx2p, dx3p, dx4p, dx5p, ...
                      dx6p, dx7p, dx8p, dx9p, dx10p, dx11p];
output(1).integrand = W1.*u1.^2 + W2.*u2.^2;

% U1 = sum(W1.*u1.^2)
% U2 = sum(W2.*u2.^2)
%-------------------------------------------------%
%--- Phase 2 -------------------------------------%
%-------------------------------------------------%
x1      = input.phase(2).state(:,1);
x2      = input.phase(2).state(:,2);
x3      = input.phase(2).state(:,3);
x4      = input.phase(2).state(:,4);

x6      = input.phase(2).state(:,6);
x7      = input.phase(2).state(:,7);
x8      = input.phase(2).state(:,8);
x9      = input.phase(2).state(:,9);
```

```
x10     = input.phase(2).state(:,10);
x11     = input.phase(2).state(:,11);

u1      = input.phase(2).control(:,1);
u2      = input.phase(2).control(:,2);

taup2     = input.phase(2).time + taudelay;


%---- Calculate Ground Effect --------------------%
z = max(hh+10.*R.*x4,hh);
% Hayden
kg = (0.9926 + 0.03794.*(z./(2*R)).^-2).^(2/3);
kg(z>(2*R)) = 1;

%---- Calculate Washout --------------------------%
% fw = max(1-x2./x2w,0);
% fw = 1;
fw = 1-1/x2w.*abs(x2);
fw(x2>x2w) = 0;
%---- Aerodynamic Model---------------------------%
% Calculate Johnson's mux = Vx_rotor/vh, muz = -Vz_rotor/vh
% updated with revised equations for mu
mux = 0.4472./(x3.*sqrt(x8)).*(x2.*cos(x9) + x1.*sin(x9));
muz = 0.4472./(x3.*sqrt(x8)).*(x2.*sin(x9) - x1.*cos(x9));

lam_i_J = ones(length(x1),1);

for m = 1:length(x1)
% Call getInducedVelocityVRS.m to obtain lambda_i
lam_i_J(m) = getInducedVelocityVRS3(mux(m),muz(m),useVRS);
end

% Regular mu, and lambda_i
mu  = 0.01./x3.*(x2.*cos(x9) + x1.*sin(x9));
lam_i = 0.0224.*lam_i_J.*(x8).^.5;

%---- End Aerodynamic Model----------------------%


%---- Calculate Power Requred -------------------%
% Main Rotor
Cp_mr = ki.*lam_i.*x8./1000 + p1.*(1+4.65.*mu.^2) + p2.*mu.^3 - p3.*x1./x3.^3;
Pr_mr = 1/M.*p4.*Cp_mr.*x3.^3;
% Tail Rotor
Pr_tr = (1-1.2*mu).*1./Mtr.*sqrt(Pr_mr.^3./(pt.*x3.^3));

% Total P_req including accessories and losses
Pr = (Pr_mr./etaMR+Pr_tr./etaTR+HPacc.*550)./etaCBOX;
pr1 = Pr./Poei;

%--- Calculate Governed Engine Power Available --%
p2g = min(pr1 - G1.*(x3-1),1);
p2g(p2g<0) = 0;

%---- Calculate Derivatives ----------------------%
dx1p = g0 - m0.*x3.^2.*x8.*(kg-fv.*fw).*cos(x9) - ...
       f0z.*x1.*sqrt(x1.^2+x2.^2);
dx2p =      m0.*x3.^2.*x8.*(kg-fv.*fw).*sin(x9) - ...
       f0x.*x2.*sqrt(x1.^2+x2.^2);
dx3p = i0./x3.*(x6+x7-pr1);
dx4p = -0.1.*x1;
dx5p = 0.1.*x2;
dx6p = -k1.*x6;
dx7p = zeros(length(x1),1);
for nn = 1:length(x1)
    if taup2(nn) > .1548
    dx7p(nn) = k2.*(p2g(nn)-x7(nn));
    end
```

```
end
dx8p  = tref.*x10;
dx9p  = tref.*x11;
dx10p = tref.*u1;
dx11p = tref.*u2;
output(2).dynamics = [dx1p, dx2p, dx3p, dx4p, dx5p, ...
                      dx6p, dx7p, dx8p, dx9p, dx10p, dx11p];

output(2).integrand = W1.*u1.^2+W2.*u2.^2;
% U1 = sum(u1.^2)/100000
% U2 = sum(u2.^2)/10000

%-------------------------------------------------%
% End Function:  GPOPSAH1ZCont10.m    %
%-------------------------------------------------%
```

D.5.3 GPOPS-II® Endpoint File

```
%-------------------------------------------------------%
% Begin Function:  GPOPSAH1ZEnd6.m
%-------------------------------------------------------%
function output = J13(input)

Wt = input.auxdata.Wt;
WL = input.auxdata.WL;

% Define Phase 1 initial and final state
tau0{1} = input.phase(1).initialtime;
tauf{1} = input.phase(1).finaltime;
xf{1} = input.phase(1).finalstate;

% Define Phase 2 initial and final state
tau0{2} = input.phase(2).initialtime;
x0{2} = input.phase(2).initialstate;
tauf{2} = input.phase(2).finaltime;

% Event Group 1:  Linkage Constraints Between Phases 1 and 2
output.eventgroup(1).event = [x0{2}-xf{1},tau0{2}-tauf{1}];

% Event Group 2:  Event that enforces phase 2 time > phase 1 time
output.eventgroup(2).event = [tauf{1}-tau0{1},tauf{2}-tau0{2}];

% Define the Initial Cost
phi = input.phase(1).initialstate(4);

% Define the Running Cost
L = input.phase(1).integral + input.phase(2).integral;

% phi
% weightedtauf = Wt*tauf{2}
% L
% weightedrunningcost = WL*L

% Objective Function
output.objective = phi + Wt*tauf{2} + WL*L;
% output.objective = phi + W*tauf{2};

%-------------------------------------------------------%
% End Function:  GPOPSAH1ZEnd6.m
%-------------------------------------------------------%
```

Appendix E. Required Aircraft Parameters

The following inputs are required for the algorithm.

Table E.1. Required Aircraft Parameters

Gross weight
Nominal rotor angular velocity
Rotor polar moment of inertia
Rotor solidity ratio
Main rotor blade average drag coefficient
Main rotor diameter
Tail rotor diameter
Root cutout (if applicable)
Tail rotor moment arm
Main rotor hub height
Equivalent horizontal flat plate drag area
Equivalent vertical flat plate drag area
Equivalent vertical flat plate drag area under rotor disc
Power required for accessories
Gearbox efficiency factors
Single engine power rating for OEI flight

Table E.2. Parameters used to Match Dynamic Model to Flight Test Data

Estimate for engine control unit gain
Estimates for engine time constants
Estimate for airspeed at which rotorwash no longer impacts majority of fuselage
Estimate for main rotor efficiency
Estimate for tail rotor efficiency
Estimate for induced velocity correction factor
Estimate for flat plate drag correction factor
Estimate for good engine response delay

Bibliography

[1] A. V. Rao, "A Survey of Numerical Methods for Optimal Control," in *2009 AAS/AIAA Astrodynamics Specialist Conference, AAS Paper 09-334*, Pittsburgh, PA, 2009, pp. 533–541.

[2] W. Johnson, "Helicopter Optimal Descent and Landing After Power Loss," NASA Ames Research Center, Tech. Rep. NASA-TM-73244, 1977.

[3] Commander, Naval Air Systems Command, "NATOPS Flight Manual Navy Model AH-1Z Helicopter," 2014.

[4] W. Johnson, *Rotorcraft Aeromechanics*, 1st ed. New York, NY: Cambridge University Press, 2013.

[5] W. J. Hanley and G. Devore, "An Analysis of the Helicopter Height Velocity Diagram Including a Practical Method for its Determination," Tech. Rep., 1968.

[6] R. J. Pegg, "An Investigation of the Helicopter Height-Velocity Diagram Showing Effects of Density and Gross Weight," Langley Research Center, Langley Station, VA, Tech. Rep., 1968.

[7] R. Studwell, "Helicopter Dynamic Performance Program Volume II - User's Manual USARTL- TR-79-27B," Sikorsky Aircraft Division, United Technologies Corporation, Tech. Rep., 1980.

[8] D. E. Kirk, *Optimal Control Theory.* Mineola, NY: Dover Publications, 1970.

[9] A. E. Bryson, *Dynamic Optimization.* Menlo Park, CA: Addison Wesley Longman, Inc., 1999.

[10] J. T. Betts, "Survey of Numerical Methods for Trajectory Optimization," *Journal of Guidance, Control and Dynamics*, no. 21, 1997.

[11] P. E. Gill *et al.*, "SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization," *SIAM Journal on Optimization*, no. 4, 2002.

[12] L. R. Balkanyi and O. F. Spurlock, "DUKSUP - A High Thrust Trajectory Optimization Code," in *Proceedings of the AIAA/AHS/ASEE Aerospace Design Conference*, Washington, DC, 1993.

[13] A. Miele *et al.*, "Sequential Gradient-Restoration Algorithm for Optimal Control Problems with Non-differential Constraints," *Journal of Optimization Theory and Applications*, vol. 28, no. 2, 1974.

[14] R. D. Russell and L. F. Shampine, "A Collocation Method for Boundary Value Problems," *Numerische Mathematik*, no. vol 19, pp. 13–36, 1972.

[15] A. Y. Lee, "Optimal Landing of a Helicopter in Autorotation," Ph.D. dissertation, Stanford University, 1985.

[16] Y. Okuno *et al.*, "Analytical Prediction of Height-Velocity Diagram of a Helicopter Using Optimal Control Theory," *Journal of Guidance, Control, and Dynamics*, vol. 14, no. 2, pp. 453–459, 1991.

[17] R. T. N. Chen and Y. J. Zhao, "Optimal Trajectories for the Helicopter in One-Engine-Inoperative Terminal-Area Operations, NASA TM 110400," NASA Ames Research Center, Tech. Rep., 1996.

[18] E. B. Carlson and Y. J. Zhao, "Prediction of Tiltrotor Height-Velocity Diagrams Using Optimal Control Theory," *Journal of Aircraft*, vol. 40, no. 5, pp. 896–905, 2003.

[19] C. L. Bottasso *et al.*, "Optimization of Critical Trajectories for Rotorcraft Vehicles," in *American Helicopter Society 60th Annual Forum*, 2004, pp. 816–835.

[20] E. N. Bachelder *et al.*, "Improved Method for Evaluating OEI Height-Velocity Boundaries," in *American Helicopter Society Vertical Lift Aircraft Design Conference*, San Francisco, California, 2006, pp. 1–13.

[21] B. L. Aponso *et al.*, "Evaluation of a Rotorcraft Autorotation Training Display on a Commercial Flight Training Device," *Journal of the American Helicopter Society*, vol. 52, no. 2, pp. 123–133, 2007.

[22] E. B. Carlson *et al.*, "H-1 Upgrades Height-Velocity Diagram Development Through Flight Test and Trajectory Optimization," in *American Helicopter Society 62nd Annual Forum Proceedings*, Phoenix, Arizona, 2006, pp. 729–742.

[23] W. Johnson, "Model for Vortex Ring State Influence on Rotorcraft Flight Dynamics," NASA Ames Research Center, Moffett Field, CA, Tech. Rep., 2005.

[24] A. Y. Lee *et al.*, "Optimal Landing of a Helicopter in Autorotation," in *American Institute of Aeronautics and Astronautics 13th Atmospheric Flight Mechanics Conference*, Williamsburg, VA, 1986, pp. 533–541.

[25] A. A. Jhemi, "Optimal Flight of a Helicopter in Engine Failure," Ph.D. dissertation, University of Minnesota, 1999.

[26] E. B. Carlson, "An Analytical Methodology for Category A Performance Prediction and Extrapolation," in *American Helicopter Society 57th Annual Forum*, 2001, pp. 1–14.

[27] M. A. Patterson and A. V. Rao, "GPOPS-II: A MATLAB Software for Solving Multiple-Phase Optimal Control Problems Using hp-Adaptive Gaussian Quadra-

ture Collocation Methods and Sparse Nonlinear Programming," *ACM Transactions on Mathematical Software*, no. 1, 2014.

[28] D. Garg *et al.*, "Direct Trajectory Optimization and Costate Estimation of Finite-Horizon and Infinite-Horizon Optimal Control Problems Using a Radau Pseudospectral Method," *Computational Optimization and Applications*, no. vol 49, issue 2, 2011.

[29] E. Carlson and J. Keane, "AH-1Z Flight Test Report for Height-Velocity Demonstration," Bell Helicopter Textron, Tech. Rep., 2006.

[30] S. Viswanathan and W. Salter, "Basic Structural Design Criteria for the AH-1Z U.S. Marine Corps Tactical Helicopter," Bell Helicopter Textron, Tech. Rep. 449, 2009.

[31] D. Sigl, "AH-1Z Substantiation Data Report, Flight Manual Performance, and Photographic Records of Equipment," Bell Helicopter Textron, Tech. Rep., 2007.

[32] R. Prouty, *Helicopter Performance, Stability and Control.* Malabar, FL: Krieger Publishing Company, 2001.

[33] W. Johnson, *Helicopter Theory.* Princeton, NJ: Princeton University Press, 1980.

[34] J. G. Leishman, *Principles of Helicopter Aerodynamics*, 2nd ed. New York, NY: Cambridge University Press, 2000.

[35] J. Kennedy, "Particle Swarm Optimization," *Encyclopedia of Machine Learning, Springer*, pp. 760–766, 1992.

[36] J. H. Holland, "Genetic Algorithms," *Scientific American, vol. 267*, no. 1, pp. 66–72, 1992.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
22 March 2018	Master's Thesis	1 April 2017 — 22 March 2018

4. TITLE AND SUBTITLE

ANALYTICAL DETERMINATION OF A HELICOPTER HEIGHT VELOCITY DIAGRAM

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Harris, Michael, Maj, USMC

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENY-MS-18-M-260

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AIR VEHICLE ENGINEERING DEPARTMENT 4.3
Attn: Steven Woods, Bldg. 2187
Naval Air Warfare Center Aircraft Division
48110 Shaw Road, Suite 1320
Patuxent River, MD 20670-1907

10. SPONSOR/MONITOR'S ACRONYM(S)

NAVAIR

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

A helicopter height velocity (HV) diagram was analytically constructed using optimal control techniques. A three degree of freedom, point-mass, dynamic model was developed and validated with flight test data. An induced velocity calculation was incorporated which addressed the affects of vortex ring state. The problem was posed as an open final time, constrained initial state, constrained final state problem, with the objective function as a weighted sum of the initial altitude and quadratic controls. This formulation was solved using direct pseudo-spectral collocation and adaptive mesh refinement as implemented by the GPOPS-II® software suite. Proper adjustment of path constraints was crucial in achieving solutions which were comparable with flight test data. Results compare favorably with flight test data and previous analytical HV diagrams.

15. SUBJECT TERMS

Height Velocity, Dynamic Optimization, Optimal Descent

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Donald L. Kunz, AFIT/ENY
U	U	U	UU	162	19b. TELEPHONE NUMBER *(include area code)* (937) 255-3636 x4548; donald.kunz@afit.edu